VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

NÁVRH DÍLČÍ ČÁSTI INFORMAČNÍHO SYSTÉMU
DESIGN OF AN INFORMATION SYSTEM PART

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Marián Kaššák**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Bernard Neuwirth, Ph.D., MSc**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA PODNIKATELSKÁ

Zadání diplomové práce

Ústav:	Ústav informatiky
Student:	**Bc. Marián Kaššák**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Informační management
Vedoucí práce:	**Ing. Bernard Neuwirth, Ph.D., MSc**
Akademický rok:	2018/19

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává diplomovou práci s názvem:

Návrh dílčí části informačního systému

Charakteristika problematiky úkolu:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Vlastní návrhy řešení
Závěr
Seznam použité literatury
Přílohy

Cíle, kterých má být dosaženo:

Cílem diplomové práce je návrh a implementace řešení na zvýšení bezpečnosti a efektivnosti správy účtů a hesel o klientech ve společnosti. Součástí diplomové práce bude provedení analýzy současného stavu, ze které bude návrh řešení a jeho následná implementace vycházet. Návrh řešení se zaměří především na identifikaci slabých míst stávajícího řešení a jejich eliminaci.
Práce bude obsahovat zhodnocení přínosů pro společnost, jehož součástí bude i ekonomické zhodnocení.

Základní literární prameny:

BASL, Josef a Roman BLAŽÍČEK. Podnikové informační systémy: Podnik v informační společnosti. 3. vyd. Praha: Grada, 2012. 323 s. ISBN 978-80-247-4307-3.

DOSTÁL, Petr, Karel RAIS a Zdeněk SOJKA. Pokročilé metody manažerského rozhodování. 1. vyd. Praha: Grada, 2005. 168 s. ISBN 80-247-1338-1.

GÁLA, Libor, Jan POUR a Zuzana ŠEDIVÁ. Podniková informatika. 3. vyd. Praha: Grada Publishing, 2015. 240 s. ISBN 978-80-247-5457-4.

MOLNÁR, Zdeněk. Efektivnost informačních systémů. 2. vyd. Praha: Grada, 2001. 179 s. ISBN 80-2470-087-5.

SODOMKA, Petr a Hana KLČOVÁ. Informační systémy v podnikové praxi. 2. vyd. Brno: Computer Press, 2010. 504 s. ISBN 978-80-251-2878-7.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2018/19

V Brně dne 28.2.2019

L. S.

doc. RNDr. Bedřich Půža, CSc. ředitel	doc. Ing. et Ing. Stanislav Škapa, Ph.D. děkan

Abstrakt

Diplomová práca sa zaoberá analýzou spoločnosti so zameraním na dátovú bezpečnosť. Návrh je zameraný na elimináciu bezpečnostných hrozieb v zabezpečení dát. Riešením je nový spôsob prihlasovania zamestnancov do CMS systémov spoločnosti, na základe API autorizácie s využitím protokolu OAuth 2.0.

Kľúčové slová

informačný systém, dátová bezpečnosť, OAuth 2.0, autorizácia, API, CMS systém

Abstract

The diploma thesis deals with the analysis of the company and the focus on data security. Based on the company analysis and requirements is designed a new system of employee authorization in CMS systems of the company. The design of the employee authorization API solution is based on the OAuth 2.0 protocol.

Key words

information system, data security, OAuth 2.0, authorization, API, CMS system

Bibliografická citácia

KAŠŠÁK, Marián. *Návrh dílčí části informačního systému* [online]. Brno, 2019 [cit. 2019-05-12]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/117733. Diplomová práce. Vysoké učení technické v Brně, Fakulta podnikatelská, Ústav informatiky. Vedoucí práce Bernard Neuwirth.

Čestné prehlásenie

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 10. května 2019

…….

podpis studenta

Poďakovanie

Touto cestou by som rád poďakoval Ing. Bernardovi Neuwirthovi, Ph.D., MSc za jeho vynaložený čas a rady, ktoré mi ochotne poskytoval pri vypracovávaní diplomovej práce. Taktiež by som rád poďakoval spoločnosti za umožnenie tvorby, konzultácie a potrebné podklady na realizáciu práce. V neposlednom rade ďakujem svojej rodine za umožnenie štúdia a podporu popri ňom.

OBSAH

ÚVOD

Informačný systém je nástroj na zefektívnenie a automatizáciu každodenných procesov v spoločnosti. Každý informačný systém je špecifický, slúži na rôzne účely a musí spĺňať parametre vyplývajúce z potrieb užívateľov. Jedným z nevyhnutných predpokladov úspešnosti spoločnosti je riadenie rizík, a teda aj rizík z oblasti dátovej bezpečnosti.

Zabezpečenie elektronických informácií je dôležitou súčasťou aktivít každej organizácie. Spoločnosť sa musí vysporiadať so stále novými zraniteľnosťami, hrozbami a rizikami, pred ktorými musí svoje informačné aktíva chrániť. Nesmie existovať slobodný prístup k aktívam spoločnosti ku ktorým však okrem zamestnancov často potrebujú prístup aj klienti. Systémy často spracovávajú citlivé informácie o klientoch, ktorých narušenie integrity by mohlo znamenať kritické problémy pre spoločnosť.

CIEĽ A METODIKA PRÁCE

V prvej časti práce sú popísane teoretické východiská, ktoré sa týkajú danej témy. Pri analýze súčasného stavu boli ohodnotené aktíva spoločnosti, spolu s ich hrozbami a pravdepodobnosťou ich naplnenia. Výsledná matica ukázala najzraniteľnejšie miesta v bezpečnosti spoločnosti, ktorých naplnenie by mohlo spôsobiť kritické problémy.

Hlavným cieľom práce je zvýšenie dátovej bezpečnosti v spoločnosti elimináciou najväčších bezpečnostných slabín. Návrh na zmenu systému prihlasovania zamestnancov do CMS systémov spoločnosti eliminuje niekoľko rizík spojených s neoprávneným prístupom k dátam.

CMS systémy spracovávajú okrem obsahu webových stránok aj informácie odoslané z webových formulárov. Objednávky, registrácie a kontaktné formuláre spracovávajú citlivé informácie o užívateľoch. Neoprávnený prístup k týmto informáciám by mohol spôsobiť kritické problémy pre spoločnosť. Je teda nevyhnutné dbať na bezpečnosť prístupu k týmto informáciám a takisto mať definované presné pravidlá pri práci s nimi.

Návrh riešenia musí prihliadať na podmienky spoločnosti. V závere budú popísané prínosy riešenia, jeho ekonomické náklady a odporúčania pre spoločnosť.

1. TEORETICKÉ VÝCHODISKÁ

Kapitola zameraná na teoretické východiská, z ktorých vychádza analytická a návrhová časť diplomovej práce.

1.1. Informačný Systém (IS) a Informačné Technológie (IT)

Informačný systém (IS) - identifikovateľný funkčný celok, zabezpečujúci systematické zhromažďovanie, spracovávanie, uchovávanie a sprístupňovanie informácií. IS integruje informačnú základňu (dáta), technické a programové vybavenie, finančné prostriedky a procedúry (procesy, predpisová základňa, manuály) a pracovníkov (správci, užívatelia) (1).

Účel informačného systému môžeme brať ako zaistenie správnych informácií na správnom mieste v správny čas, pre jeho vybraných užívateľov (4).

IS je tvorený ľuďmi, ktorí prostredníctvom dostupných technologických prostriedkov a stanovenej metodiky spracovávajú podnikové dáta a vytvárajú z nich informačnú a znalostnú bázu organizácie slúžiacu k riadeniu podnikových procesov, manažérskemu rozhodovaniu a správe podnikovej agendy (2).

Informačné technológie (IT) - technika, ktorá sa zaoberá spracovaním informácií, t.j. predovšetkým výpočtová a komunikačná technika (hardware), a jej programové vybavenie (software). Vlastnosť kvality IS môžeme definovať ako systém, ktorý spĺňa zadefinované požiadavky (1).

Obecné ukazovatele kvality systémov:

- **funkčnosť** (plní funkcie, na ktoré bol implementovaný),
- **vzhľad** (estetická stránka),
- **spoľahlivosť a údržba** (stály výkon, úžitkovosť a ľahká údržba),
- **trvanlivosť** (doba životnosti),
- **bezpečnosť** (zabezpečenie dát) (1).

1.2. Bezpečnosť IS

Hrozba je akcia alebo udalosť, ktorá môže spôsobiť, že informácia alebo zdroje spracovávania informácií budú zámerne alebo náhodne stratené, modifikované, kompromitované, stanú sa nedostupnými alebo budú inak negatívne ovplyvnené k ujme spoločnosti (4).

Riziko - vyjadruje mieru nebezpečia, že sa uplatní hrozba a dôjde k nežiadúcemu výsledku vedúcemu k vzniku škody (4).

Zraniteľnosť - je slabé miesto IS pre bezpečnosť hodnoteného aktíva (4).

Bezpečnostné opatrenie - prostriedok slúžiaci k zmierneniu pôsobenia hrozby (elimináciu), zníženia zraniteľnosti alebo dopadu hrozby (4).

Analýza rizík - slúži k odhadu strát, ktoré môžu vzniknúť pôsobením hrozieb na IS a dáva prehľad o stupni nebezpečnosti jednotlivých hrozieb, slabých miest (zraniteľnosti) hodnoteného IS a rizikách, ktorým je hodnotený IS vystavený (4).

Bezpečnosť dát je ochranou proti nepovolenému prístupu, prenosu, zmenám alebo zničeniu. Autentifikácia a autorizácia predstavujú kľúčové úlohy pri vytváraní a udržovaní komunikácie medzi klientom a službou. Ich úlohou je zabezpečiť, aby boli vykonané len také operácie, ktoré sú povolené (18).

Autentifikácia je proces overenia používateľa (overenie jeho identity), teda zistenie, o koho ide. Overenie môže byť napríklad podľa kombinácie prihlasovacieho mena a hesla, biometrie (otlačok prsta) alebo rozpoznanie hlasu. Autentifikácia môže byť vykonaná aj iným zariadením, respektíve službou (18).

Autorizácia je proces overenia oprávnení pre danú úlohu, teda či autentifikovaný používateľ (systém) má oprávnenie vykonať úlohu, o ktorú žiada. Napríklad vstúpiť do systému (18).

Dôvernosť predstavuje potrebnú úroveň miery utajenia v každom okamžiku, kedy dochádza k spracovaniu dát a je zaistená prevencia ich neautorizovaného vyzradenia. Úroveň dôvernosti je nutné udržať pri uchovávaní dát v systémoch, pri ich prenose a po doručení adresátovi. Situácie vedúce k porušeniu dôvernosti sú napríklad v prípade útoku, kedy sú prekonané mechanizmy zaisťujúce dôvernosť, ale aj v situácií kedy užívateľ (úmyselne alebo chybou) vyzradí citlivú informáciu (nezašifrovaním odosielanej správy, podľahnutie sociálnemu inžinierstvu) (4).

Integrita je udržaná, ak sú dáta presné, so zaručeným obsahom pričom sú zabezpečené opatrenia proti ich neautorizovanej zmene. Hardwarové, softwarové a komunikačné prostriedky musia pracovať tak, aby dáta uchovávali a spracovávali správne a presne, prenášali ich do požadovaného cieľa bez nežiadúcich zmien (4).

Dostupnosť zapríčinenie nedostupnosti je snahou útočníka ovplyvniť produktivitu, alebo daný systém vyradiť z prevádzky. Systém musí mať dátovou kapacitu dimenzovanú tak, aby v definovanom čase poskytoval dostatočný výkon (4).

1.3. Aplikačná vrstva

Aplikačná vrstva je siedmou a poslednou vrstvou referenčného OSI modelu. Jej funkciou je sprostredkovať a poskytovať služby aplikáciám. V aplikačnej vrstve pracujú sieťové aplikácie a ich protokoly patriace do aplikačnej vrstvy. Aplikačná vrstva Internetu obsahuje mnoho protokolov, napríklad protokol **HTTP** (je popísaný v nasledujúcej podkapitole), SMTP (zaisťuje prenos e-mailových správ) a FTP (poskytuje prenos súborov medzi dvoma koncovými systémami). Protokol aplikačnej vrstvy definuje, ako si procesy aplikácií, bežiacich na rôznych koncových systémoch, medzi sebou predávajú správy. Protokol aplikačnej vrstvy definuje:

- Typy správ, ktoré sú vymieňané (napríklad správy s požiadavkou a správy s odpoveďou).
- Syntax rôznych typov správ (napríklad pole v správe a to, ako sú vymedzené).
- Sémantiku týchto polí (význam informácií v poli).
- Pravidlá, ktoré určujú, kedy a ako proces správy odosiela a reaguje (15).

1.4. HTTP protokol

HTTP (HyperText Transfer Protocol) je protokol **aplikačnej** vrstvy a je implementovaný na dvoch miestach: v klientskom programe a v programe serveru pracujúcich na rôznych koncových systémoch. Komunikujú spolu výmenou správ HTTP, pričom HTTP protokol definuje štruktúru týchto správ a spôsob, akým si klient a server správy vymieňajú. **Webový**

prehliadač realizuje stranu klienta a **webové servery**, ktoré implementujú serverovú stranu HTTP (klient-server architektúra), obsahujúcu webové objekty. Webová stránka sa skladá z objektov (napríklad HTML súbor, JavaScript súbor, JEPG obrázok...), ktoré sú adresovateľné pomocou jedinečnej URL adresy. Každá URL adresa má dve zložky: názov serveru, na ktorom je objekt uložený a názov cesty k objektu. Príklad:

https://www.example.sk/img/picture.jpeg

kde hostiteľ je *www.example.sk* a názov cesty k súboru je */img/picture.jpeg*.

Protokol HTTP definuje, ako weboví klienti žiadajú o webové stránky z webových serverov a ako servery prenášajú webové stránky ku klientom. Ak užívateľ požaduje webovú stránku, prehliadač pošle správu so žiadosťou HTTP na objekty stránky. Server žiadosť príjme a odpovie správou HTTP, obsahujúcu objekty (16).

HTTP používa protokol TCP ako svoj základný transportný protokol (miesto UDP). Klient HTTP najskôr inicializuje TCP spojenie so serverom. Ak je spojenie nadviazané, prehliadač (klient) a server spracujú TCP prístup prostredníctvom svojich soketových rozhraní. Klient odosiela do svojho soketového rozhrania správy s HTTP požiadavkami a prijíma zo svojho soketového rozhrania správy s HTTP odpoveďami. Protokol TCP poskytuje protokolu HTTP službu spoľahlivého prenosu dát, to znamená že každá správa s požiadavkou HTTP odoslaná procesom klienta nakoniec dorazí neporušená na server. Rovnako HTTP odpoveď zo strany serveru v nezmenenom stave príde na stranu klienta (16).

HTTP je **bezstavový** protokol - ak klient požiada (v intervale niekoľko sekúnd) dvakrát o rovnaký objekt, server odpovie dvakrát rovnakým objektom (nepamätá si prvú žiadosť) (18).

1.4.1. Formáty správ HTTP

Špecifikácie protokolu HTTP (RFC 1945; RFC 2616) zahrňujú definíciu formátu správ HTTP. Sú dva typy HTTP správ: **žiadosť** a **odpoveď** (18).

A.) Žiadosť HTTP

Príklad HTTP žiadosti môže vyzerať nasledovne:

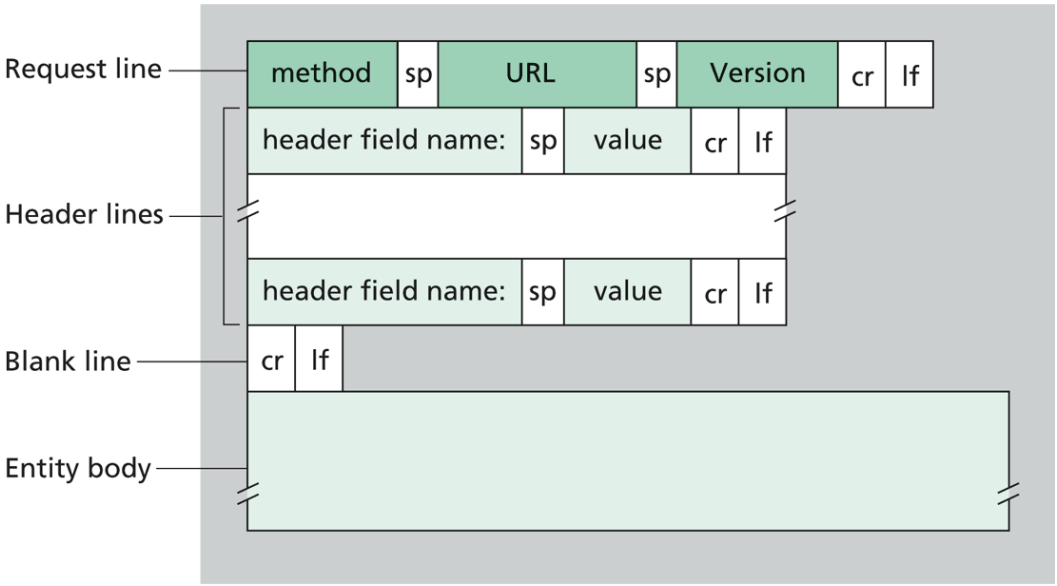
```
GET /adresat/page.html HTTP/1.1

Host: www.example.sk

Connection: close

User-agent: Mozilla/5.0

Accept-language: sk
```

Správa je v bežnom ASCII texte. Prvý riadok žiadosti sa nazýva **riadok žiadosti**, nasledujúce riadky sa nazývajú **riadky záhlavia**.

Riadok žiadosti má tri časti: metóda, pole URL adresy a verzia HTTP. Metóda žiadosti môže byť: *GET, POST, HEAD, PUT* a *DELETE*. V danom príklade si prehliadač metódou *GET* žiada o objekt, ktorý je určený v poli URL adresy: */adresat/page.html*.

Riadok záhlavia *Host*, určuje počítač, na ktorom je objekt umiestnený. *Connection: close* určuje dočasné spojenie. User-agent určuje prehliadač odosielajúci požiadavku a Accept-language určuje preferovanú verziu objektu (ak takú server nemá, pošle predvolený) (19).

Obecný formát HTTP žiadosti vyzerá nasledovne:



Obrázok 1: Obecný formát správy HTTP žiadosti (zdroj: 19)

Telo správy je s metódou *GET* prázdne, ale používa sa s metódou *POST*. Metódou *POST* užívateľ požaduje webovú stránku zo serveru, ale konkrétny obsah webovej stránky závisí na tom, čo užívateľ zadal napr. do poľa formuláru (napríklad na vyhľadanie záznamu s ID 123). Ak by sa v tomto prípade použila metóda *GET*, telo žiadosti by bolo prázdne a zadané dáta by boli vložené priamo do URL napríklad: *www.example.sk/search?id=123* (19).

Metóda *PUT* sa využíva pre publikovanie na webe. Umožňuje nahrať objekt na určitú adresu na konkrétnom serveri. Metóda *DELETE* umožňuje užívateľovi odstrániť objekt z webového serveru. Ak server obdrží žiadosť metódy *HEAD*, odpovie správou HTTP pričom vynechá požadovaný objekt (19).

B.) Odpoveď HTTP

Príklad nasledujúcej HTTP odpovede, môže predstavovať odpoveď na predchádzajúci príklad HTTP žiadosti:

```
HTTP/1.1 200 OK

Connection: close

Date: Tue, 09 Apr 2019 15:44:02 GMT

Server: Apache/2.2.3

Last-Modified: Tue, 09 Apr 2019 15:44:02 GMT

Content-Length: 6821

Content-Type: text/html

(data data data .....)
```

Odpoveď má tri časti: počiatočný **stavový riadok**, šesť riadkov **záhlavia** a **telo** správy.

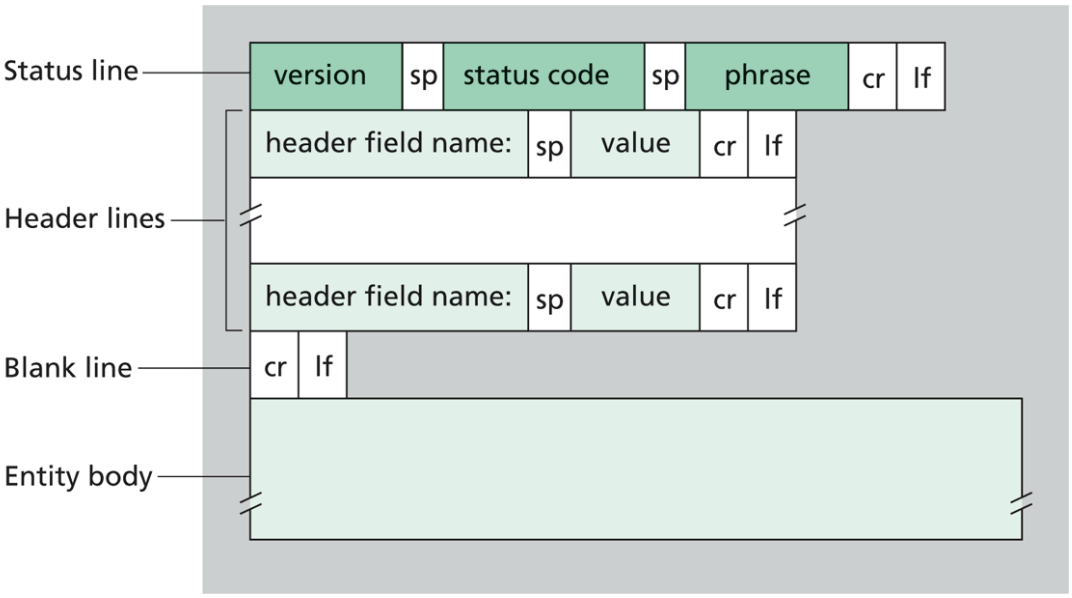
Stavový riadok v tomto príklade ukazuje, že server využíva verziu HTTP/1.1 a že je všetko v poriadku - server bol dostupný a posiela požadovaný objekt (19)

Záhlavie správy obsahuje niekoľko riadkov.

- *Connection: close* znamená že po odoslaní správy chystá ukončiť pripojenie TCP.
- *Date:* označuje dátum a čas, kedy server vytvoril a odoslal odpoveď HTTP. Je to čas, kedy server načítal objekt zo svojho súborového systému, vložil objekt do správy a tú odoslal.
- *Server:* znamená, že správy vygeneroval webový server Apache. Tento riadok je analogický s riadkom záhlavia User-agent: v správe žiadosti.
- *Last-Modified:* označuje dátum a čas vytvorenia alebo poslednej zmeny objektu, je dôležitý pre uloženie objektu do medzi pamäte, a to ako u klienta tak aj webových proxy serveroch.
- *Content-Length:* udáva počet bajtov v odoslanom objekte.
- *Content-Type:* znamená, že objekt v tele správy je HTML text.

Telo správy je hlavnou častou, obsahuje požadovaný objekt (reprezentovaný reťazcom `data data data`...).

Obecný formát odpovede HTTP vyzerá nasledovne:



Obrázok 2: Obecný formát správy s odpoveďou HTTP (zdroj: 19)

Stavový kód súvisiace frázy ukazujú výsledok žiadosti:

- 200 OK: žiadosť bola úspešná a vrátená odpoveď obsahuje objekt.
- 301 Moved Permanently: Požadovaný objekt bol trvalo presunutý. Nová URL adresa je uvedená v záhlaví správy s odpoveďou Location:
- 400 Bad Request: Obecný chybový kód v prípade, ak server nerozumie žiadosti.
- 404 Not Found: Požadovaný dokument na danom serveri neexistuje.
- 505 HTTP Version Not Supported: Požadovaná verzia protokolu HTTP nie je na danom serveri podporovaná.

1.5. Útoky na webové aplikácie

Útočník po odhalení zraniteľnosti webovej aplikácie môže túto aplikáciu napadnúť. V tejto kapitole budú rozobraté najčastejšie príklady útokov na webové aplikácie.

1.5.1. Cross-site request forgery (CSFR)

Snaha útočníka vykonať užívateľa akciu bez jeho vedomia. Príkladom je snaha podstrčiť odkaz, ktorý upravuje správanie aplikácie do ktorej je obeť prihlásená vo svojom prehliadači. Podstatou CSFR útoku je odoslanie HTTP žiadosti z jednej webovej aplikácie do inej pod identitou napadnutého užívateľa (12).

Zabezpečenie pre CSFR útokmi je možné kontrolou HTTP Refferera v poli hlavičky žiadosti. Alternatívou je definovaním reťazca znakov, ktorý sa predá v žiadosti a vedia ho indetifikovať obe strany. Z tejto hodnoty je možné zistiť, z akej stránky bola žiadosť odoslaná (12).

1.5.2. Injection a SQL Injection

Injection využíva nesprávne ošetreného vstupu od klienta pri zostavovaní dotazov a parametrov do iných systémov. SQL Injection je najčastejším príkladom - zlá obsluha parametrov pri tvorbe SQL dotazov (13).

Príklad nesprávneho použitia premennej *param*

*String query = "select * from tabulka where field = '"+param+"'";*

Pri zadaní hodnoty param napríklad: *"'or 1=1'"*, môže dosiahnuť na dáta z celej tabuľky, pretože dotaz bude vyzerať nasledovne:

*select * from tabulka where field = ' ' or 1=1*

prípadným doplnením subselektov alebo joinov môže prečítať dáta aj iných tabuliek.

Obranou pred útokom typu SQL Injection je nezostavovať SQL dotazy pomocou jednoduchého skladania reťazcov (13).

1.5.3. Krádež session

Pretože je HTTP bezstavový protokol, ale potrebuje určitú formu stavu prenášať, používa sa mechanizmus session. Užívateľovi je pri prvom prístupe vytvorený session a je pridelená jeho identifikácia pomocou cookie, prípadne parametrom GET. V prípade získania identifikátor sa dá session zneužiť (13).

Pri použití cookie, je nutné zaistiť kontrolu IP adresy a dobu platnosti. Pri využití ukladania session do URL, je nutné zaistiť, aby URL nebolo zaslané žiadnemu inému serveru ako HTTP Referer (z ktorej stránky užívateľ prišiel). V tomto prípade je nutné zabrániť načítavaniu externých zdrojov (CSS, obrázky, scripty) z iných serverov, a pri odkazovaní na iný server je potrebné presmerovanie užívateľa (pomocou refresh v HTML stránke užívateľa presmerovanie na novú stránku, presmerovaním pomocou protokolu HTTP sa nezmení referer) (13).

1.5.4. Cross Sire Scripting (XSS)

V tomto prípade sa útočník snaží do kódu HTML vygenerovaného serverom vložiť skript, ktorý sa potom v prehliadači užívateľa vykoná pri načítaní stránky.

Existujú dva typy XSS útoku: **perzistentné** a **reflektované** (12).

Pri perzistentnom XSS útoku je snahou vložiť skript do dátového úložiska aplikácie. Webový prehliadač následne vykoná uložení script pri načítaní stránky. Dátovým úložiskom je vo väčšine prípadov relačná databáza, ktorú je možno využiť na ukladanie vstupov od používateľa. Ich následné načítanie v HTML dokumente je hrozbou práve perzistentného XSS útoku (12).

Pri reflektovanom XSS útoku sa skript nevkladá do dátového úložiska, ale vkladá sa jednorazovo s príchodom požiadavky zo strany užívateľa na webový server (*GET* alebo *POST* metóda). V prípade *GET* metódy sú parametre a hodnoty priamo súčasťou URL adresy, a teda je možnosť vytvoriť odkaz, ktorý ako parameter predáva webovej aplikácií útočný skript. Príklad predania scriptu v parametry metódou *GET*:

www.example.sk/hladat.php?dotaz=<script>alert(/XSS/);</script>

Pri využití POST metódy, sa parametre a ich hodnoty nepredávajú v URL adresách, ale sú obsahom tela HTTP žiadosti. Útočník nie je schopný teda vytvoriť útočný odkaz. V tomto prípade útočník pripraví webovú stránku, ktorá odošle pripravené dáta vrátane útočného skriptu. Užívateľ, ktorý nasleduje odkaz na takto pripravenú stránku, ktorá ho ďalej presmeruje na napádanú webovú stránku. Pričom sa napádanej stránke predajú POST parametre z útočníkovej stránky spolu s útočným skriptom (12).

Pre zabezpečenie webovej aplikácie proti XSS útokom, je nutné escapovať nedôveryhodné dáta pred ich vložením do obsahu HTML dokumentu, prípadne atribútu. Ochranou je teda ošetrenie obsahu predaného *GET* parametru pred jeho vložením, prípadne použitím v JavaScripte (12).

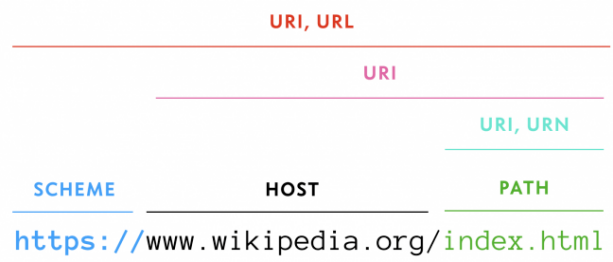
1.5.5. Sociálne inžinierstvo

Ide o spôsob manipulácie ľudí za účelom prevedenia určitej akcie alebo získania určitej informácie. Termín je používaný vo význame podvodu, respektíve podvodného jednania za účelom získania utajených informácií organizácie alebo prístupu do informačných systémov firmy. Vo väčšine prípadov útočník neprichádza do osobného kontaktu s obeťou. Príkladom je Phishing, čo je podvodná technika používaná k získaniu citlivých údajov (heslo). Princípom je rozoslanie emailových správ, ktoré sa tvária ako oficiálne žiadosti, napríklad vedenia spoločnosti a vyzývajú adresáta k zadaniu požadovaných údajov na odkazovanú stránku. Táto stránka môže napríklad napodobňovať systém, ktorý užívateľ pozná a do ktorého sa bežne prihlasuje. Zadaním mena a hesla sú prístupy odcudzené (11).

Pretexting je príkladom jednej z metód sociálneho inžinierstva. Ide o vytvorenie a využitie vymysleného scenáru s cieľom presvedčiť obeť k uskutočneniu požadovanej akcie alebo k získaniu potrebnej informácie. Pretexting sa dá využiť napríklad pri vydávaní sa za kolegu, zamestnanca alebo spoločnosť.

1.6. URI vs URL

V práci sú používané výrazy URI a URL preto je v krátkosti rozobratý ich rozdiel. URI je skratka pre jednotný identifikátor zdroja a URL znamená jednotný lokátor zdroja (10).



Obrázok 3: URI vs URL rozdiel (zdroj: 10)

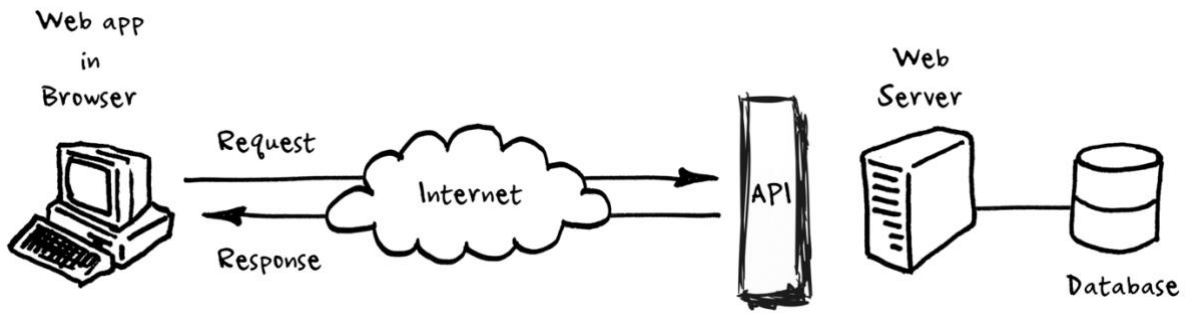
URL je lokátor zdroja. URI je identifikátor. Sú teda prepojené, to znamená, že všetky adresy URL sú identifikátory URI. Ale nefunguje to opačne (10).

1.7. API

Application **P**rogramming **I**nterface - umožňuje rôznym aplikáciám komunikovať cez internet. Ide o rozhranie software-to-software (teda nie užívateľské rozhranie). API sa dá chápať ako "čašníka" medzi "klientom" (žiadosť našej webovej aplikácie) a "kuchyňou" teda externým serverom z ktorého požadujeme dáta (20).

1.7.1. Web API

Web API špecifikujú, ako medzi sebou komponenty komunikujú s využitím HTTP protokolu, pričom ich odpoveď je vrátená vo formáte JSON alebo XML. Klient pritom nepotrebuje poznať akú procedúru má volať na servery. Výhodou web API je flexibilita. Klient a poskytovateľ API sú na sebe absolútne nezávislé, a teda môžu využívať rôzne programovacie jazyky (PHP, Java, Ruby...) pre ich časť implementácie (20).



Obrázok 4: Princíp fungovania API (Zdroj: 14)

Príkladom môže byť proces vyhľadania leteniek. Ak let vyhľadávame na stránke konkrétnej leteckej spoločnosti (Ryanair, Emirates ..). Komunikujeme len cez stránku leteckej spoločnosti a dotazujeme sa na ich databázu.

V prípade ak využívame službu vyhľadávač cestových spojení (Skyscanner,Kiwi..). Služba komunikuje s verejne dostupnými API viacerých leteckých spoločností. Odpoveď je najčastejšie vo formáte JSON.

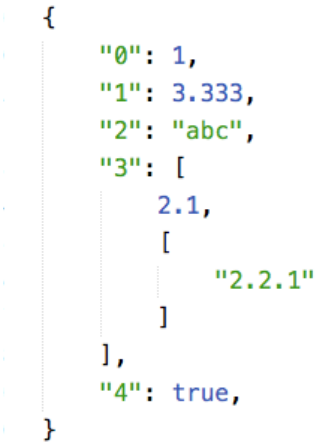
1.8. JSON

JavaScript Object Notation je spôsob zápisu dát (dátový formát) nezávislý na počítačovej platforme, určený pre prenos dát, ktoré môžu byť organizované v poliach alebo agregované v objektoch. Vstupom je ľubovoľná dátová štruktúra (číslo, reťazec, boolean, objekt alebo z nich zložené pole), výstupom je vždy reťazec. Zložitosť hierarchie vstupnej premennej nie je teoreticky nijak obmedzená (12).

Príklad JSON (v jazyku PHP):

```
$a=array(1, 3.333, "abc", array(2.1, array("2.2.1")), true);
echo json_encode($a, JSON_PRETTY_PRINT);
```

Výpis kódu vo formáte JSON vyzerá nasledovne:



Obrázok 5: Výpis vo formáte JSON (zdroj: vlastné spracovanie)

JSON je obecný formát a slúži na prenos dát v ľubovoľnom programovacom alebo skriptovacom jazyku a je čitateľný aj pre človeka.

1.9. JSON Web Token (JWT)

Je štandard (RFC 7519) založený na JSON na vytváranie prístupových tokenov. Server napríklad vygeneruj prístupový token s oprávnením "prihlásený ako admin" a poskytne ho klientovi, ktorý ho použije na preukázanie prihlásenia. Token je podpísaný súkromným kľúčom jednej strany (serveru), takže druhá strana je schopná overiť legitimitu tokenu. JWT tokeny sa typicky využívajú na odovzdanie identity overených používateľov medzi poskytovateľom identity a poskytovateľom služieb (21).

1.9.1. Štruktúra JWT

Tri časti JWT sú zakódované pomocou kódovania Base64url.

Tabuľka 1: Štruktúra JSON Web Tokenu (zdroj: 21)

Header	{
	"alg" : "HS256",
	"typ" : "JWT"
	}
		Označuje algoritmus, ktorý bol použitý na generovanie podpisu. HS256 označuje podpis tokenu pomocou HMAC-SHA256

Payload	<pre>{
	"loggedInAs" : "admin",
	"iat" : 1422779638
	}</pre>
		"iat" (issued at) - definuje jeden zo siedmych štandartných prvkov pola. Identifikuje čas vytvorenia (NumericDate).
		"loggedInAs" je vlastný prvok.

Signature	<pre>HMAC-SHA256(
	base64urlEncoding(header) + '.' +
	base64urlEncoding(payload),
	secret
	)</pre>
		Podpis je vypočítaný Base64url kódovaním Headeru a Payloadu. Výsledný reťazec prechádza cez kryptografický algoritmus špecifikovaný v hlavičke (HMAC-SHA256).

1.10. OAuth 2.0

OAuth 2 je protokol, ktorý umožňuje aplikáciám získať obmedzený prístup k používateľským kontám v službe HTTP. Ide o autorizačný protokol, a teda sa zaoberá informáciami "kto prideluje prístup komu". Autentifikácia je prvý krok procesu protokolu (8).

Cieľom je poskytnúť bezpečnú autentizáciu a autorizáciu oproti API rôznych služieb, a to pre desktopové, mobilné a webové aplikácie. Prevádzkovateľom služieb, ktorý OAuth identity poskytuje, dáva možnosť zdieľať dáta užívateľov z ich účtu (meno, email, fotka...) bez toho, aby užívateľ prezradil svoje heslo komukoľvek ďalšiemu (8).

1.10.1. Všeobecný priebeh protokolu

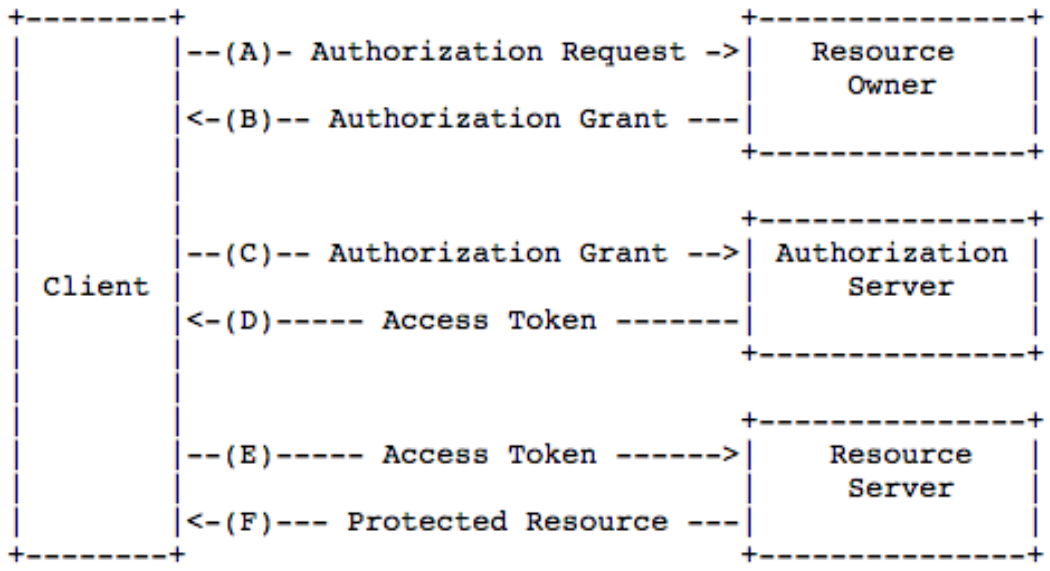
Na obrázku je znázornený všeobecný priebeh komunikácie protokolu OAuth 2.0. Sú definované 4 role:

Vlastník zdroja je *používateľ*, ktorý autorizuje aplikáciu na prístup k svojmu účtu. Prístup aplikácie k účtu používateľa je obmedzený na „rozsah“ udelenej autorizácie (napr. Prístup na čítanie alebo zápis - parameter *scope*).

Klient je aplikácia, ktorá chce pristupovať k účtu *používateľa*. Predtým, ako tak môže urobiť, musí byť autorizovaný užívateľom a autorizácia musí byť validovaná API.

Zdrojový a **Autorizačný** server: **API**

Zdrojový server je hostiteľom chránených účtov používateľa a autorizačný server overuje jeho identitu a potom vydáva prístupové tokeny do aplikácie. **API služby teda prestavujú kombináciu služieb zdrojových a autorizačných serverov.**



Obrázok 6: Všeobecný priebeh protokolu OAuth 2.0 (Zdroj: 8)

A. Klient (webová aplikácia) požiada o autorizáciu vlastníka.

B. Klient dostane autorizačný grant predstavujúci oprávnenie vlastníka zdroja.

C. Klient s predloženým autorizačným grantom požiada o autentifikáciu na autorizačnom serveri.

D. Autorizačný server autentifikuje klienta a overí udelenie autorizácie, a ak je platný, vydá prístupový token.

E. Klient požaduje chránený zdroj zo zdrojového serveru, pričom sa autentifikuje s predloženým prístupovým tokenom.

F. Zdrojový server overí prístupový token, v prípade platnosti, obslúži žiadosť (8).

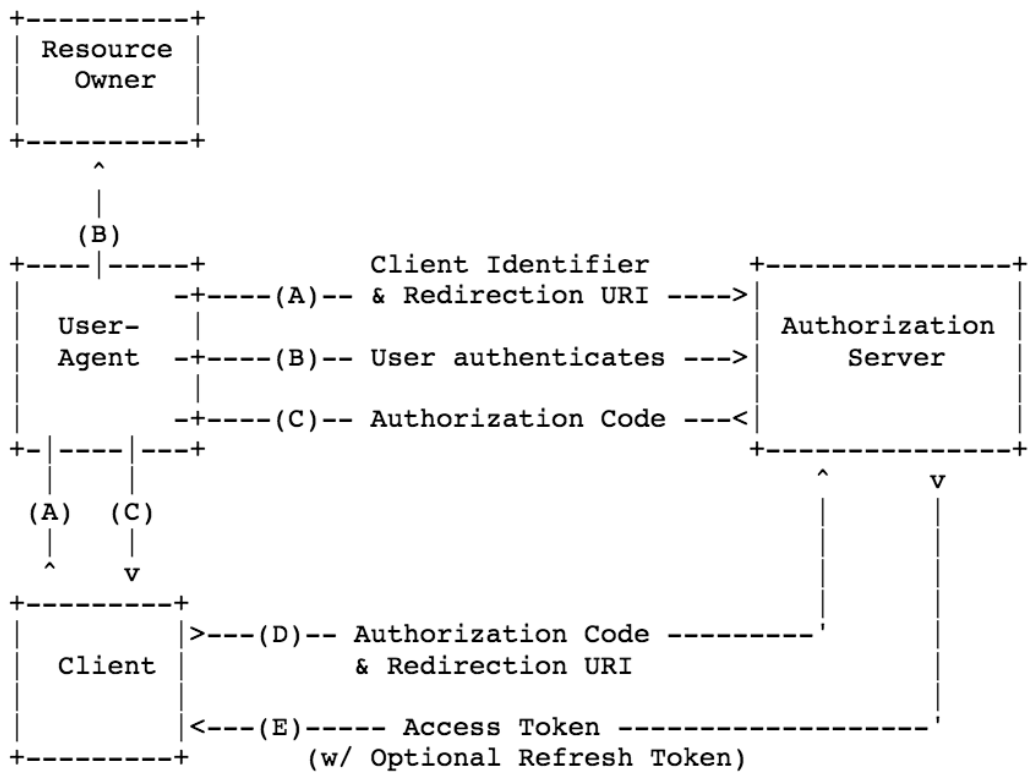
Špecifikácia OAuth 2.0 opisuje štyri typy **grantov** („metód“) na získanie prístupového tokenu (čo predstavuje oprávnenie používateľa na prístup klienta k ich údajom), ktoré možno použiť:

- Authorization code (udelenie autorizačného kódu)
- Implicit
- Resource owner password credentials (udelenie poverenia vlastníka zdroja)
- Client credentials (udelenie poverení klienta)

Detailnejšie bude rozobraný prvý typ spôsobu získania autorizácie - *Authorization Code Grant.*

1.10.2. Authorization Code Grant

Ide o metódu získania prístupového tokenu (access token). Ide o tok presmerovaní, klient teda musí byť schopný interakcie s rozhraním vlastníka zdroja (zvyčajne webovým prehliadačom) a schopný prijímať prichádzajúce požiadavky (cez presmerovania) z autorizačného serveru.



Obrázok 7: Authorization Code Flow (zdroj: 8)

V zobrazenom priebehu udelenie autorizačného kódu, sú kroky (A), (B) a (C) rozdelené na dve časti podľa priechodu cez *User-Agent* (webový prehliadač). Jednotlivé časti priebehu sú popísané v návrhovej časti na konkrétnom riešení.

1.10.3. Žiadosť na autorizáciu

(A / B) Klient vytvorí URI žiadosť pridaním parametrov na autentifikáciu v prostredí poskytovateľa API.

Tabuľka 2: Parametre žiadosti o autorizáciu (Zdroj: vlastné spracovanie podľa 8)

response_type	REQUIRED	Hodnota musí byť nastavená na "*code*"
client_id	REQUIRED	Jedinečný identifikátor klienta
redirect_uri	OPTIONAL	Musí byť definovaná minimálne jedna hodnota
scope	OPTIONAL	Rozsah povolení
state	RECOMMENDED	Reťazec na prevenciu pred falšovaním identity žiadostí

Napríklad klient smeruje vo webovom prehliadači na nasledujúcu HTTP žiadosť:

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz
   &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

Autorizačný server validuje žiadosť, aby zabezpečil platnosť požadovaných parametrov. Ak je žiadosť platná, autorizačný server pristúpi k autorizácii užívateľa.

1.10.4. Odpoveď na autorizáciu

(C) Ak používateľ (vlastník zdroja) potvrdí autorizáciu k jeho účtu, autorizačný server poskytne autorizačný kód (code) a doručí ho klientovi pridaním nasledujúcich parametrov do komponentu dotazu adresy presmerovania (redirect_uri), s využitím "application/x-www-form-urlencoded" formátu (8).

code	REQUIRED	Autorizačný kód vygenerovaný autorizačným serverom. Je doporučené nastavenie maximálne 10 miútového životného cyklu parametru. Klient nesmie použiť autorizačný kód viac ako jeden krát. Autorizačný kód je viazaný na ID klienta a redirect_uri.
state	REQUIRED (if..)	Je vyžadovaný ak bol definovaný v predchádzajúcej žiadosti klienta na autorizáciu.

Klient musí ignorovať nerozpoznané parametre odpovede. Veľkosť reťazca autorizačného kódu nie je definovaná, preto klient musí očakávať rôzne dĺžky reťazca.

Chybová odpoveď

Ak žiadosť zlyhá kvôli chýbajúcej, neplatnej alebo nezhodujúcej sa adrese presmerovania (redirect_uri), alebo v prípade neplatnej identifikácií klienta, autorizačný server informuje užívateľa (resource owner) o chybe, ale nesmie automaticky presmerovať na redirect_uri.

Ak užívateľ zamietne žiadosť o povolenie prístupu aplikácie k požadovaným informáciám, alebo zlyhá z iného dôvodu (iného ako chýbajúca alebo neplatná URI), autorizačný server informuje klienta pridaním nasledujúceho parametru do komponentu dotazu:

Tabuľka 3: Chybové hlášky na žiadosť o autorizáciu (zdroj: vlastné spracovanie podľa 8)

invalid_request	chýbajúci alebo neplatný povinný parameter
unauthorized_client	klient nebol autorizovaný k danej žiadosti
access_denied	užívateľ, alebo autorizačný server zamietol žiadosť
unsupported_response_type	nepodporovaná metóda k získaniu autorizačného kódu
invalid_scope	hodnota parametru scope sa nezhoduje alebo nebola zadaná
server_error	HTTP žiadosť nebolo možné doručiť

1.10.5. Žiadosť o prístupový token

(D) Klient vytvorí HTTP žiadosť na získanie prístupového tokenu (*access_token*) odoslaním nasledujúcich parametrov:

Tabuľka 4: Parametre žiadosti o prístup (zdroj: vlastné spracovanie podľa 8)

grant_type	REQUIRED	hodnota musí byť nastavená na "*authorization_code*"
code	REQUIRED	autorizačný kód (*code*) prijatý z autorizačného serveru
redirect_uri	REQUIRED	musí sa zhodovať *s redirect_uri* parametrom zadanom pri autorizačnej žiadosti
client_id	REQUIRED	jednoznačný identifikárot klienta

Príklad HTTP žiadosti klienta o prístupový token:

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
```

1.10.6. Odpoveď pre prístupový token

(E) Ak je žiadosť na prístupový token platná a autorizovaná, autorizačný server vydá prístupový token. HTTP úspešnej odpovede obsahuje parametre:

Tabuľka 5: Parametre odpovede o prístup (zdroj: vlastné spracovanie podľa 8)

access_token	REQUIRED	prístupový token vydaný autorizačným serverom
token_type	REQUIRED	typ tokenu
expires_in	RECOMMENDED	životnosť tokenu v sekundách. Hodnota "3600" nastaví životnosť prístupovému tokenu na hodinu, od vygenerovania odpovede.
refresh_roken	OPTIONAL	pre využitie rovnakého autorizačného grantu k prítupovému tokenu
scope	REQUIRED	rozsáh práv k prístupu informáciam o účte užívateľa

Parametre sú súčasťou tela HTTP odpovede využitím "application/json" typom média. Parametre sú súčasťou JSON štruktúry. Parametre a hodnoty reťazcov sú JSON reťazcom. Číselné hodnoty odpovedajú JSON číselným hodnotám. Poradie parametrov nie je dôležité. Podmienkou je, aby autorizačný server obsahoval HTTP "Cache-Control" odpoveď s hodnotou "no-store" a to pre každú odpoveď obsahujúcu tokeny, prístupy alebo citlivé dáta. Takisto "Pragma", v hlavičke odpovede musí mať hodnotu "no-cache" (8).

Príklad úspešnej odpovede:

```
HTTP/1.1 200 OK
   Content-Type: application/json;charset=UTF-8
   Cache-Control: no-store
   Pragma: no-cache

   {
     "access_token":"2YotnFZFEjr1zCsicMWpAA",
     "token_type":"example",
     "expires_in":3600,
     "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
     "example_parameter":"example_value"
   }
```

2. ANALÝZA SÚČASNÉHO STAVU

Kapitola sa zaoberá analýzou súčasného stavu spoločnosti. Analýza sa zameriava na aktíva spoločnosti a ich hrozby. Cieľom analýz je odhalenie bezpečnostných slabín spoločnosti a z toho dôvodu v práci nebude uvedený jej názov.

2.1. Základná informácia o organizácii

Spoločnosť začala svoje služby poskytovať už v roku 1998 a od svojho začiatku sa zaoberala poskytovaním komplexných služieb tvorby internetových prezentácií, multimediálnych riešení, internetovou reklamou a realizáciou zákazkového softwaru. Organizačná štruktúra je funkcionálna v rámci jednotlivých divízií. Za celkový chod spoločnosti zodpovedajú dvaja majitelia, ktorí sú zároveň aj konatelia spoločnosti. Spoločnosť funguje v troch divíziach:

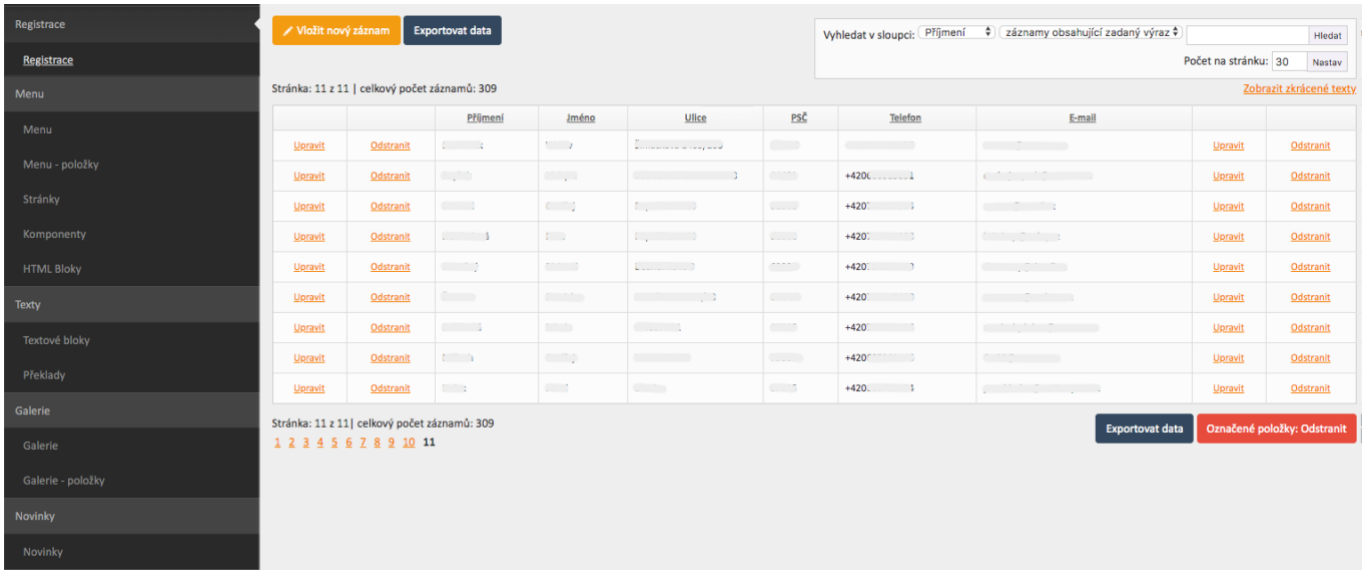
- web-development,
- mobilné aplikácie,
- hardwarová divízia.

Celkovo vo všetkých divíziách pracuje do 40 zamestnancov. Každá divízia má svojho lídra. Líder preberá zodpovednosť nad vecami spadajúcimi do jeho kompetencie. Práca je zameraná na divíziu webových aplikácií, návrhom vlastného riešenia je nutné pokryť všetky divízie a zamestnancov.

2.2. CMS systém (redakčný systém / administrácia)

Je vlastným produktom spoločnosti, ktorý je poskytovaný k webovým stránkam. CMS systém (redakčný systém / administrácia) je webová aplikácia na vytváranie a správu obsahu webu a to bez znalosti programovania. Administrácia pracuje okrem dát webovej stránky aj s dátami, ktoré boli odoslané z webu (rezervácie, registrácie, správy z formulárov). Klient po prihlásení do administrácie vie tieto informácie spracovávať.

Spoločnosť má v súčasnosti v prevádzke zhruba 60 samostatne fungujúcich CMS systémov, na rôznych webových hostingoch. Každá administrácia je konfigurovaná podľa požiadaviek a potrieb klienta.



Obrázok 8: CMS systém - registrácie (zdroj: vlastné spracovanie)

V ukážke z prostredia CMS systému je zobrazená práca aplikácie s dátami odoslaných z webu. Ide o spracovanie citlivých údajov pri registrácií užívateľov z webovej stránky.

2.2.1. Prihlasovanie do CMS systémov

Prihlasovanie do systémov je tvorené z dvoch prístupových bodov. **Prístup zamestnancov** spoločnosti (programátori, kóderi, projektový tím, grafici) a **prístup klienta**.

Klientovi je vytvorený prístupový účet, respektíve účty podľa potrieb. Prístupové údaje sú uchovávané v databázy priamo na danom webovom hostingu aplikácie. Účty klienta môžu byť s rôznym rozsahom oprávnení (admin má dosah na celý obsah webu, editor má prístup len k určitej časti). Autorizácia teda prebieha priamo na serveri kde beží webová aplikácia.

Spoločnosť má v súčasnosti desiatky klientov, ktorí využívajú ich CMS, pričom každý je na inom hostingovom priestore. Systém prihlasovania zamestnancov do administrácie webov je momentálne riešené jedným spoločným prihlasovacím účtom pre všetkých zamestnancov pre danú administráciu. Pre každý projekt (redakčný systém) je vytvorený jeden spoločný prihlasovací účet pre všetkých zamestnancov, ktorý je uložený v databázy na webovom hostingu aplikácie. Prístupové údaje sú pre každý projekt jedinečné a evidujú sa spolu s prístupovými údajmi (FTP údaje, prístupy do databáz) klienta v zdielaných kontaktoch klienta.

2.3. Systémy využívané v spoločnosti

V tejto kapitole budú popísané systémy, ktoré spoločnosť využíva.

2.3.1. Microsoft SBS 2011

Spoločnosť využíva integrovaný serverový balík **Small Business Server 2011** od spoločnosti **Microsoft**, ktorý slúži na prevádzku sieťovej infraštruktúry (intranetová správa). Ide teda o centralizovanú správu firemných stolných počítačov. Každý zamestnanec ma k dispozícií firemný stolný počítač, ktorý mu bol pridelený. Prihlasuje sa ním svojím doménovým menom a heslom.

Active Directory Domain Service (domain controller), overuje a autorizuje všetkých zamestnancov na firemných počítačoch v sieti. Ukladá informácie o účastníkov domény vrátane zariadení, overuje ich oprávnenia a definuje ich prístupové práva.

Serverový balík Small Business Server 2011 Standart poskytuje taktiež služby:

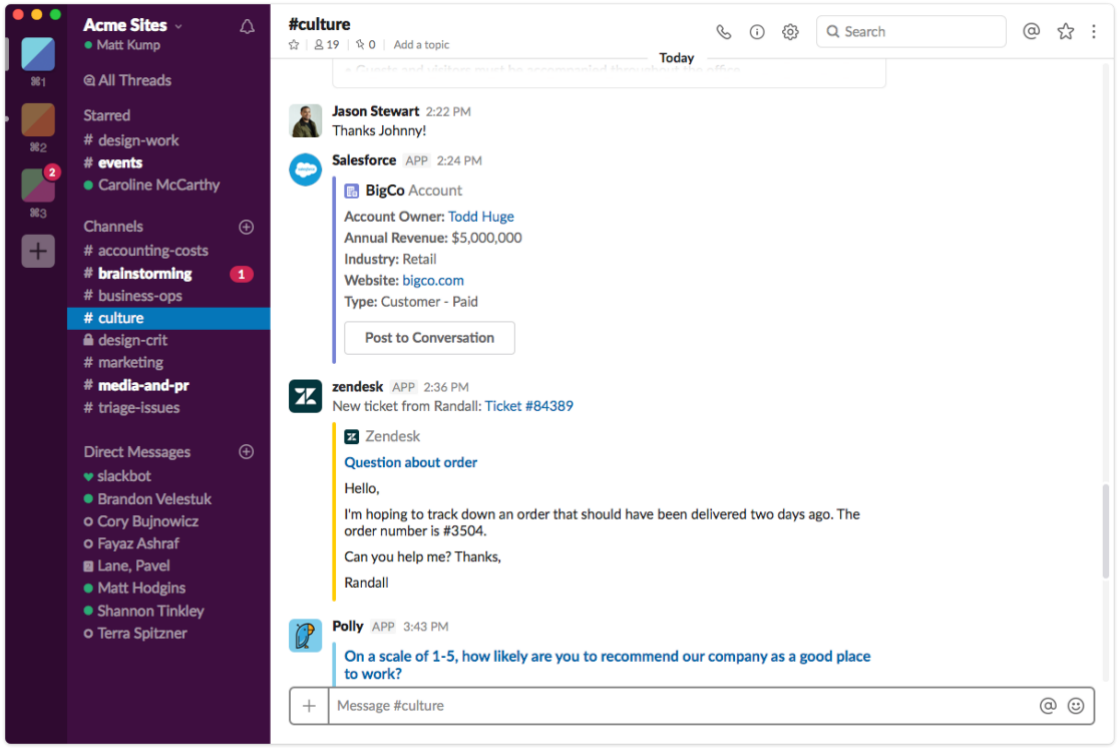
- E-mail (Microsoft Office Outlook)
- Vzdialený webový prístup (pripojenie k počítaču v kancelárií, prístup k súborom a k internému webu.
- Interný web (zdieľanie dokumentov, oznámení, kalendáru, udalosti)
- Zdieľané dokumenty (knižnice dokumentov podporujú rôzne typy dokumentov)

2.3.2. EasyRedmine

Spoločnosť na každodennú prácu využíva platformu EasyRedmine, ktorá poskytuje vytváranie úloh k danému projektu a ich následne delegovanie a spoluprácu. Ide o platformu, kde sa pracuje na základe úloh. Vo väčšine prípadov aj s predpísaným časom na trvanie prác, ktorá vychádza z dohodnutej cenovej kalkulácie projektového tímu s klientom. Úlohy spravidla padajú od projektového tímu prípadne vedúcich pracovníkov jednotlivých divízií, ktorí dozerajú na priebeh a dodržanie časových termínov. Systém takisto zabezpečuje evidenciu odpracovaného času (odpracovaný čas slúži pre výplatné pásky) a dochádzky (medzi prihlásením a odhlásením užívateľa zo systému).

2.3.3. Slack

V rámci spoločnosti zamestnanci medzi sebou komunikujú prostredníctvom platformy **Slack** (skratka od "Searchable Lot of All Conversation and Knowledge"). Ide o cloudovú sadu nástrojov a služieb pre tímovú spoluprácu (komunikáciu) v rámci konkrétneho workspace.



Obrázok 9: Prostredie platformy Slack (zdroj: 11)

Prihlasovanie do Slacku funguje na základe **workspace** (pracovného prostredia).

Workspace je v Slacku zdieľaný kanál, kde všetci členovia môžu komunikovať a spolupracovať. V sledovanej spoločnosti sa využíva jeden workspace pre všetkých zamestnancov, do ktorého sa prihlasujú svojím firemným mailom.

Bezpečnosť Slacku

Bezpečnosť je postavená na "defense in depth", t.j. bezpečnosť organizácie a dát na všetkých vrstvách.

Bezpečnostný program Slacku je v súlade s:

- **FedRAMP** (Federal Risk and Authorization Management program) - je Americký celoštátny štandardizovaný prístup k hodnoteniu bezpečnosti, autorizácií a nepretržitému monitorovaniu cloudových produktov a služieb).

- **NIST 800-171** - požiadavky, ktoré musia spĺňať nefederálne systémy a organizácie pre ochranu uchovávaných dát.

- **AICPA Trust Service Principles** (**SOC 2** a **SOC 3)**, ide o zásady dôveryhodných služieb.

- **ISO/IEC 27001** - medzinárodne platné štandardy, definujú požiadavky na systém managementu bezpečnosti informácií, predovšetkým na riadenie bezpečnosti dôvery informácií pre zamestnancov, procesy, IT systémy a stratégiu firmy.

- **ISO/IEC 27017** - štandardy, týkajúce sa bezpečnosti v cloud computingu.

- **EU/US Privacy Shield** - zásady ochrany osobných údajov pri prenose medzi EU a US.

- **CSA** (Cloud Security Aliance) - svetová organizácia, ktorá definuje zásady pre zabezpečené prostredie cloud computingu.

Ochrana dát Slacku

Snaha zabrániť neautorizovanému prístupu k dátam zákazníkov.

Tranzitné dáta - prenos medzi klientom a Slack službou, je uskutočnená s využitím silných šifrovacích protokolov. Slack podporuje najnovšie odporúčané šifrovacie balíky na šifrovanie všetkých prenosov (protokol TLX 1.2, AES256 šifrovanie, SHA2 podpisy).

Uložené dáta sú šifrované pomocou štandardu šifrovania **FIPS 140-2** (americký štandard zabezpečenia počítača používaný na schvaľovanie kryptografických modulov), ktorý sa vzťahuje na všetky typy údajov v systéme Slacku. Všetky šifrovacie kľúče sú uložené na zabezpečenom serveri v segregovanej sieti s veľmi obmedzeným prístupom. Slack zaviedol vhodné ochranné opatrenia na ochranu vytvárania, ukladanie, získavanie a zánik tajných údajov, ako sú šifrovacie kľúče.

Zabezpečenie siete a servera

Slack rozdeľuje svoje systémy do samostatných sietí, čo zvyšuje ochranu citlivých údajov. Testovacie a vývojové prostredie je oddelené od produkčného prostredia. Sieťový prístup k produkčnému prostrediu z verejných sietí je obmedzený. Dostupné sú iba tie sieťové protokoly, ktoré sú nevyhnutné pre poskytovanie služieb Slacku svojim používateľom (existuje zameranie proti DDoS útokom). Slack zaznamenáva, monitoruje a kontroluje všetky systémové dotazy a upozorňuje na potencionálne narušenia.

2.4. Identifikácia a ohodnotenie aktív

V tejto podkapitole budú analyzované aktíva spoločnosti vzhľadom na ich možné ohrozenie. Aktíva sú rozdelené do troch skupín podľa ich typu: dáta, hardware, software.

Prvým krokom je identifikácia a následne ohodnotenie. Hodnota každého identifikovaného aktíva je vypočítaná (C-I-A) vzorcom:

$$\textbf{Hodnota aktíva} = \frac{(\boldsymbol{dovernost} + \boldsymbol{integrita} + \boldsymbol{dostupnost})}{\mathbf{3}}$$

Tabuľka 6: Klasifikácia aktív (Zdroj: Vlastné spracovanie podľa: 4)

Klasifikačné kritérium	Klasifikačný stupeň
žiadny dopad na spoločnosť	1
zanedbateľný dopad	2
problémy alebo finančné straty	3
vážne problémy alebo finančné straty	4
existenčné problémy	5

Tabuľka 7: Ohodnotenie hodnoty aktív (Zdroj: Vlastné spracovanie podľa: 4)

Typ	Aktívum	Zdroj	C	I	A	H
Dáta	dáta o zamestnancoch	server, PC	5	4	4	4
	dáta o klientoch*	server, CMS	5	5	4	5
	dáta klientov**	server, CMS	5	5	4	5
	interné dáta	firemný server	4	4	4	4
Hardware	Firemný server		4	4	5	4
	Sieťové prvky		4	4	4	4
	PC		4	4	3	4
Software	Operačný systém	PC	3	4	2	3
	Databáza	Firemný server	4	4	4	4

*dáta klientov - FTP prístupy na server, prístupy do databáz pre produkčné a vývojové prostredie, prístup do redakčného systému. Tieto dáta sú uchovávané v zdieľaných kontaktoch v aplikácií Outlook.

**dáta klientov - ide o dáta odoslané prostredníctvom webovej stránky klienta (formuláre, objednávky, registrácie a pod.), ktoré sú spracovávané v CMS systémoch.

Tabuľka zobrazuje celkové hodnotenie aktív spoločnosti. Hodnota 5 značí kritickú hodnotu. Ohrozenie týchto aktív má pre spoločnosť až existenčné problémy, preto je treba aplikovať opatrenia na elimináciu hrozieb, ktoré môžu ohroziť kritické aktíva.

2.5. Identifikácia hrozieb a zraniteľností

V tejto časti sú identifikované hrozby a zraniteľnosti, ktoré môžu hroziť aktívam spoločnosti.

Tabuľka 8: Hodnotenie vzniku hrozby (zdroj: vlastné spracovanie podľa: 4)

Pravdepodobnosť hrozby	Klasifikačný stupeň
Veľmi nízka	1
Nízka	2
Stredná	3
Veľmi vysoká	4
Vysoká	5

Tabuľka 9: Zoznam hrozieb a príkladu zraniteľnosti (Zdroj: Vlastné spracovanie)

Typ	Hrozba	P	Príklad
fyzické poškodenie	poškodenie hrubou silou	1	zalomenie konektoru
	prehriatie zariadenia	1	nedostačujúce chladenie
	vznik požiaru	3	neopatrnosť s ohňom v bytovom priestore
Ohrozenie informácií	prelomenie hesla	4	úspešný útok na slabé heslo
	krádež informácií	5	sociálne inžinierstvo
	modifikácia informácií	3	útok na aplikáciu (napríklad XSS)
	vymazanie informácií	4	poškodenie úložného média
	prezradenie údajov	5	nesprávne ukončené prístupové práva
Ohrozenie funkčnosti	nesprávna manipulácia	2	ľudský faktor (neúmyselné vymazanie)
	odoprenie činností	1	nesprávne nastavenie
	chybné fungovanie	2	nesprávne nastavené zariadenie

Tabuľka 10: Zraniteľnosť aktíva na konkrétnu hrozbu (Zdroj: Vlastné spracovanie)

Zraniteľnosť		Zdroj	server, PC	server, CMS	server, CMS	server				PC	server
		Aktívum	dáta o zamestnancoch	dáta o klientoch	dáta klientov	interné dáta	firemný server	sieťové prvky	PC	Operačný systém	Databáza
		A	4	5	5	4	4	4	4	3	4
Hrozba	**T**										
poškodenie hrubou silou	1						2	2	2		
prehriatie zariadenia	1						1	1	1		
vznik požiaru	3						3	3	2		
prelomenie hesla	4		2	3	5	3					2
krádež informácií	5		2	4	5	2	1	2			
modifikácia informácií	3		2	2	3	3				2	2
vymazanie informácií	4		2	2	3	4					
prezradenie údajov	5		3	2	5	1					
nesprávna manipulácia	2		2	4	4	3	1	2		2	2
odoprenie činností	1						1	2			
chybné fungovanie	2						2	2		1	1

2.5.1. Matica rizík

V matici rizík je vypočítané riziko parametrami:

$$\mathbf{R = A * T * V}$$

- **A** - hodnota aktíva
- **T** - pravdepodobnosť hrozby
- **V** - zraniteľnosť hrozby

Podľa výsledkov v matici (maximum dosahuje hodnotu 125), bola schéma hodnotenia rozvrhnutá nasledovne:

Tabuľka 11: Ohodnotenie celkového rizika (zdroj: vlastné spracovanie podľa: 4)

Riziko	Klasifikačný stupeň
Bezvýznamné	0 - 19
Malé	20 - 39
Mierne	39 - 79
Nežiadúce	80 - 100
Kritické	101-125

V nasledujúcej matici rizík sú ohodnotené riziká s najväčším dopadom pre spoločnosť a pravdepodobnosťou. Ich naplnenie by mohlo mať kritické následky pre chod spoločnosti, a preto je nutné zapracovať na ich eliminácií, ideálne úplnom odstránení.

Tabuľka 12: Hodnoty jednotlivých rizík (Zdroj: Vlastné spracovanie)

Riziká		**Zdroj**	firemný server, PC	hosting, CMS	hosting, CMS	firemný server				PC	firemný server
		Aktívum	dáta o zamestnancoch	dáta o klientoch	dáta klientov	interné dáta	firemný server	sieťové prvky	PC	Operačný systém	Databáza
		A	4	5	5	4	4	4	4	3	4
Hrozba	**T**										
poškodenie hrubou silou	1						8	8	8		
prehriatie zariadenia	1						4	4	4		
vznik požiaru	3						36	36	24		
prelomenie hesla	4		32	60	100	48					32
krádež informácií	5		40	100	125	40	16	40			
modifikácia informácií	3		24	30	45	36				18	24
vymazanie informácií	4		32	40	60	64					
prezradenie údajov	5		60	50	125	20					
nesprávna manipulácia	2		16	40	40	24	8	16		12	16
odoprenie činností	1						4	8			
chybné fungovanie	2						16	8		6	8

Ako najväčšou kritickou hrozbou, sa ukázali **dáta o klientoch** a **dáta klientov**.

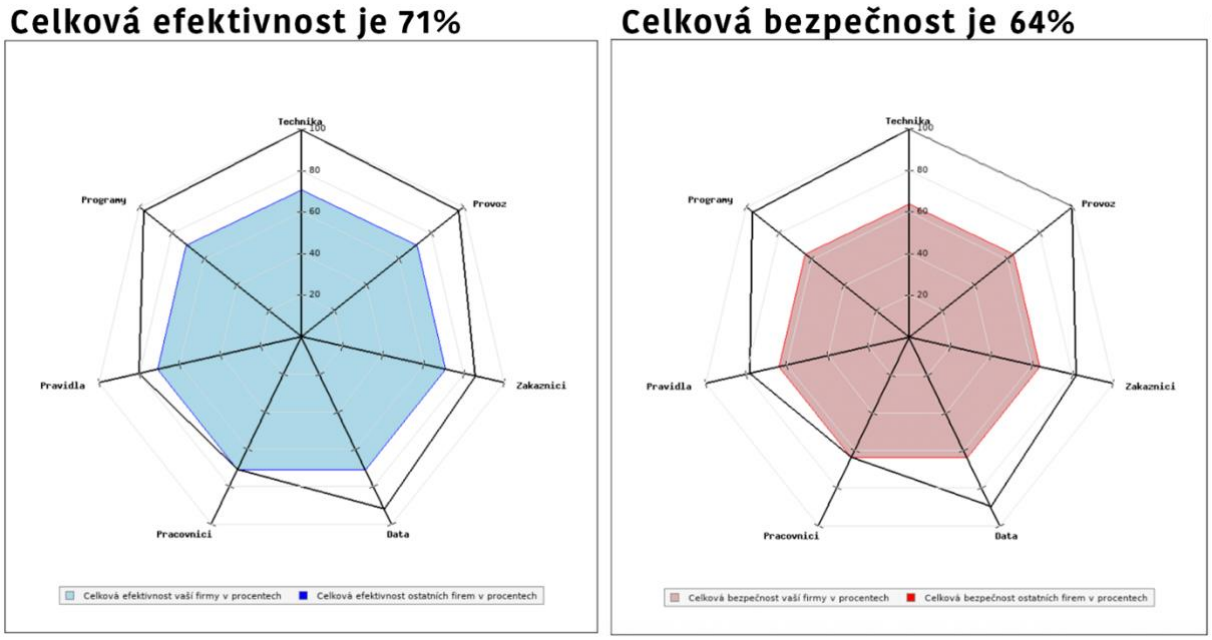
Dáta o klientoch (prístupy na FTP, databázu, vývojové prostredie) sú uchovávané a zdieľané v službe Outlook formou zdieľaných kontaktov medzi zamestnancami spoločnosti.

Dáta klientov sú údaje odoslané z webových stránok klientov (rezervácie, registrácie, objednávky), ktoré spracováva CMS systém, ktorý je produktom spoločnosti. Ide často o citlivé údaje o užívateľoch ako meno, priezvisko, adresa, v niektorých prípadoch aj rôzne splnomocnenia, objednávky, obchodné tajomstvá. Únik týchto informácií by mohli spôsobiť kritické problémy.

Najkritickejšie hrozby sú spojené s únikom dát spracovávaných v CMS systémoch. Dôvod hrozieb vychádza zo súčasného spôsobu prihlasovania zamestnancov do systémov a nesprávne ukončovanie prístupov po ukončení pracovného pomeru. Pre každý redakčný systém sa vytvorí jeden prihlasovací účet a heslo, ktorým sa do systému prihlasujú všetci zamestnanci. Z toho vychádza niekoľko bezpečnostných hrozieb. Napríklad využitie techník sociálneho inžinierstva by mohlo viesť k vyzradeniu prístupov. Prístupové mená a heslá nie sú aktualizované pri ukončení pracovného pomeru niektorého zo zamestnancov. To by bolo administratívne náročné pretože spoločnosť má v prevádzke desiatky samostatne fungujúcich CMS systémov (na rôznych hostingových priestoroch). Veľké množstvo prístupových účtov môže viesť k nesprávnemu zachádzaniu s heslami. Zamestnanec s veľkým počtom potrebných hesiel môže mať sklon k uchovávaniu hesiel v nezašifrovanej forme v PC alebo fyzicky na papieri. Bývalý zamestnanec pozná všetky prístupy a je schopný manipulovať s dátami klientov spoločnosti, vyzradiť ich konkurencií prípadne využiť vo svoj prospech.

2.6. Analýza Zefis

Zefis je online platforma, ktorá pomáha preveriť a zlepšiť fungovanie firmy, procesov a informačných systémov a overiť úroveň bezpečnosti a to aj s ohľadom na GDPR. Pomocou dotazníkov pomáha odhaliť kľúčové nedostatky a poukazuje na ich odstránenie.



Graf 1: Celková efektívnosť a bezpečnosť analýz Zefis (Zdroj: 6)

Výsledný graf pozostáva z 7 bodov: **Technika**, **Provoz**, **Zakaznici**, **Data**, **Pracovníci**, **Pravidla** a **Programy**.

Výsledok celkovej bezpečnosti dosiahol **64%.** Celková bezpečnosť je daná najslabším článkom a teda tento výsledok ukázal bezpečnostnú slabinu v spoločnosti na strane zamestnancov. Jej dôvod bude rozobratý po jednotlivých nezhodách, ktoré vyšli z analýz.

Výsledok celkovej efektívnosti **71%** je vyšší ako u bezpečnosti, snahou by pre spoločnosť však malo byť vyrovnanie jednotlivých oblastí na približne rovnakú úroveň.

V skúmanej spoločnosti vyšla oblasť zamestnancov najslabšia. Dôvod tohto výsledku ako aj jednotlivé hlavné body sú dalej rozobraté.

Oblast	Významnost	Bezpečnost	Typ	Název
Pracovníci	Vysoká	Ano	Neshoda	Přístupová práva zaměstnanců nejsou správně ukončována
Pracovníci	Vysoká	Ano	Neshoda	Nastavení přístupových práv
Zákazníci	Vysoká	Ano	Neshoda	Nejsou nastavena pravidla práce s daty zákazníků
Pracovníci	Vysoká	Ano	Neshoda	Není vytvářeno bezpečnostní povědomí pracovníků
Pracovníci	Vysoká	Ano	Neshoda	Neprobíhají periodická bezpečnostní školení uživatelů IS
Pracovníci	Vysoká	Ano	Neshoda	Nejsou aktualizovaná hesla uživatelů
Pravidla	Vysoká	Ne	Neshoda	Špatně stanovená zodpovědnost pracovníků v procesu

Obrázok 10: Efektívnosť a úroveň bezpečnosti v spoločnosti (Zdroj: 6)

Prístupové práva zamestnancov nie sú správne ukončované

Po ukončení pracovného pomeru zamestnanec nestráca prístupové práva do CMS systémov klientov. Nie sú aktualizované prístupové heslá pre účty zamestnancov. V súčasnom riešení prihlasovania je to administratívne prakticky nemožné aktualizovať heslá ku každému CMS systému.

Chýbajúca podpora pracovníkov pri práci s IS

V našom prípade ide o spoločnosť vyvíjajúcu software, a teda zamestnanci sú IT špecialisti. A preto je toto riziko znížené na minimum.

Chýbajúce alebo nesprávne dodržiavané bezpečnostné pravidlá

V súčasnom stave chýba zodpovednosť a kontrola práce s dátami klientov. Zamestnanci sa neprihlasujú pod vlastným účtom a tým sa sťažuje kontrola ich práce.

Pracovníci môžu inštalovať programy na svojich počítačoch

V prípade sledovanej spoločnosti ide o ľudí pracujúcich v IT. Riziko je znížené ich povedomí o hrozbách internetu (licencie, malware..). Každý programátor má preferované vlastné vývojové prostredie, takže inštalovať programy nie je možné v tomto prípade obmedziť.

Chýbajú bezpečnostné pravidlá IS

Pravidlá a hlavne zodpovednosť je chýbajúca. Keďže sa všetci zamestnanci prihlasujú na klientské administrácie pod jedným účtom, chýba "osobná zodpovednosť" za prácu s dátami klientov. Nie je možné dohľadať autora zmien v CMS systémoch.

2.7. Vyhodnotenie analýz

Analýza súčasného stavu spoločnosti bola zameraná na odhalenie bezpečnostných slabín v spoločnosti. V úvode kapitoly je v krátkosti predstavená spoločnosť, jej obor podnikania a jej hlavný produkt - CMS systém. Následne boli predstavené systémy, ktoré zefektívňujú prácu zamestnancov a sú využívané v rámci celej spoločnosti.

Prvým krokom k identifikácií hrozieb a zraniteľností, bolo ohodnotenie aktív spoločnosti v troch skupinách - hardware, software a dáta. V druhom kroku boli identifikované hrozby s konkrétnym možným príkladom a pravdepodobnosťou jeho naplnenia. Bola zostavená matica zraniteľností, z ktorej vyšli tri kritické hodnoty, ktoré boli spojené s ohrozením dát klientov. Výsledkom bola matica rizík, ktorá bola vypočítaná súčinom hodnoty aktíva, jeho zraniteľnosti a pravdepodobnosti naplnenia. Kritickú hodnotu dosiahlo prezradenie a ukradnutie dát klientov.

Najväčšie riziká vychádzajú zo zabezpečenia dát v CMS systémoch. Tieto systémy na riadenie obsahu, spracovávajú okrem dát webových stránok aj dáta odoslané z webových formulárov. V prípade rezervácií, objednávok a registrácii na webových stránkach ide o citlivé údaje, ktorých spracovanie a ukladanie podlieha aj legislatívnym nariadeniam. Je potrebné mať presne zadefinované pravidlá a pridelenú zodpovednosť pre prácu s citlivými údajmi.

Pri súčasnom spôsobe prihlasovania, sú najväčšie hrozby spojené s nesprávnym ukončovaním prístupových práv bývalých zamestnancov a jednotným prístupom pre všetkých zamestnancov. Rizikom je aj nedodržanie zásad uchovávania hesiel zapríčinené veľkým počtom prístupových hesiel. Spoločnosť má momentálne v prevádzke desiatky CMS systémov, a každý systém má jedinečné prístupové údaje.

Celková bezpečnosť je daná najzraniteľnejším miestom a preto elimináciou týchto bezpečnostných rizík sa zvýši bezpečnosť celej spoločnosti. Vlastný návrh riešenia bude zameraný na zmenu spôsobu prihlasovania zamestnancov do CMS systémov s cieľom zvýšenia dátovej bezpečnosti.

3. NÁVRH RIEŠENIA

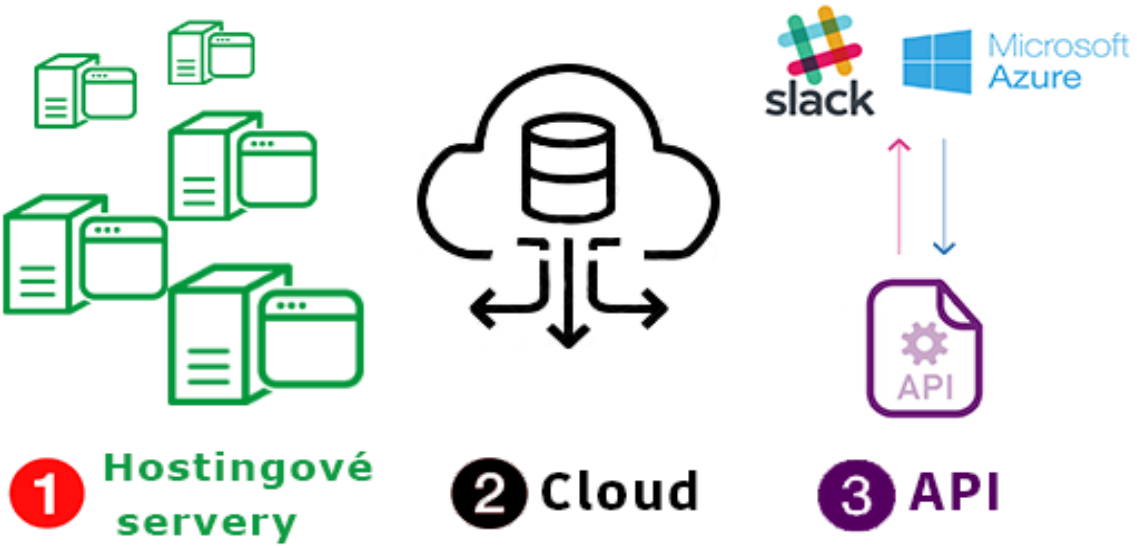
Návrhová časť vychádza z analýz spoločnosti a jej bezpečnostných rizík. Návrh riešenia má za cieľ zvýšenie bezpečnosti dát spoločnosti a to odstránením jej najväčších hrozieb. Návrh na zmenu v spoločnosti je v zmene spôsobu prihlasovania zamestnancov do CMS systémov.

3.1. Požiadavky zo strany spoločnosti

Požiadavky zo strany spoločnosti boli, aby nové riešenie neprinieslo veľkú administratívnu záťaž. To znamená aby pri odchode zamestnanca alebo príchode nového nemusel poverený správca odoberať prístupové práva, respektíve ich pridávať manuálne. Spoločnosť sa chce vyhnúť tomu, aby museli najať správcu, ktorý by zodpovedal za delegovanie prístupov na CMS systémy alebo pridať túto zodpovednosť na niektorého zo súčasných zamestnancov.

3.2. Možnosti realizácie

Cieľom je nahradiť súčasné riešenie prihlasovania zamestnancov do CMS systémov. Aktuálne má spoločnosť v prevádzke niekoľko desiatok samostatne fungujúcich redakčných systémov na rôznych hostingových serveroch. Pričom pre každú administráciu je vytvorený jeden login a heslo pre všetkých zamestnancov. Sú tri spôsoby prihlasovania, ktorých implementácia by zvýšila bezpečnosť dát oproti súčasnému spôsobu:



Obrázok 11: Možnosti realizácie (zdroj: vlastné spracovanie)

1 Prvý návrh prihlásenia zamestnancov do redakčného systému predstavuje autorizáciu priamo na hostingových serveroch. Prihlasovacie údaje uložené priamo na hostingovom serveri, kde prebieha aj autorizácia prístupu. Pričom by bol vytvorený samostatný prihlasovací účet pre každého zamestnanca, ktorý potrebuj prístup do danej administrácie.

Klady	Zápory
+ Vlastný prihlasovací účet zamestnanca	- Nejednotná správa účtov
+ Prideľovanie prístupov na konkrétne projekty	- Veľkí počet prístupov pre zamestnanca

2 Druhý spôsob predstavuje riešenie autorizácie zamestnancov na centrálnom (firemnom) serveri, kde sú evidované účty zamestnancov. Po úspešnom prihlásení sa poverenie na prístup prepošle na danú administráciu.

Klady	Zápory
+ Vlastný prihlasovací účet zamestnanca	- Administratívna záťaž
+ Centralizovaná správa	- Nové prístupy pre zamestnanca
+ Jednotný prístup zamestnanca do systémov	

3 Tretí spôsob riešenia je API autorizácia zamestnancov s ich existujúcim účtom tretej strany. Zamestnanec by bol autorizovaný na základe účtu v existujúcom systéme, ktorý sa v spoločnosti využíva.

Klady	Zápory
+ Vlastný prihlasovací účet zamestnanca	- Funkčnosť závisí na externom systéme
+ Automatizovaná správa prístupov	
+ Jednotný prístup zamestnanca	
+ Delegovanie prístupov	
+ Nevzniknú nové prístupové údaje	
+ Bezpečnosť a efektívnosť práce	

3.2.1. Výber realizácie

Prvý spôsob prináša vysokú administratívnu záťaž v prípade odchodu zamestnanca (respektíve príchodu nového). Správca musí odobrať prístupové práva zo všetkých hostingových serverov, na ktorých mal zamestnanec vytvorený prístup. Negatívom je aj vyššie riziko úspešného útoku na aplikáciu. Každý hostingový server, ktorý uchováva prístupové údaje zvyšuje riziko prelomenia bezpečnosti. V prípade rôznych prihlasovacích údajov zamestnanca do jednotlivých administrácií hrozí ich nesprávne uchovávanie (ukladanie v nezašifrovanej podobe v PC a pod.)

Druhý spôsob s centralizovanou správou prístupov znižuje administratívne zaťaženie. V prípade odchodu zamestnanca, správca odoberie prístupové práva len na jednom serveri. Možnosť delegovania prístupov pre jednotlivé administrácie je možný len v prípade konfigurácie aplikácie, čo prináša ďalšiu administratívu.

Tretí spôsob nevyžaduje pri odchode zamestnanca žiadny zásah správcu. Pri ukončení pracovného pomeru, zamestnanec stráca prístup do systémov využívaných v spoločnosti. Tým stráca možnosť prístupu do CMS systémov. Správnym nastavením podmienok autorizácie je možné delegovanie prístupov. Zamestnancom nepribudnú nové prihlasovacie údaje.

Pre návrh vlastného riešenia bola **zvolená tretia možnosť** - API autorizácia zamestnancov na strane externého systému.

3.3. Výber API poskytovateľa

Na základe porovnaní možných riešení bol zvolený návrh na zmenu prihlasovania zamestnancov na základe autorizácie účtom externého systému. V analytickej časti práce sú spomenuté systémy, ktoré sa využívajú v rámci celej spoločnosti.

Systém pre autorizáciu zamestnancov musí spĺňať dve podmienky:

- musí poskytovať verejné API pre autorizáciu zamestnancov
- musí byť využívaný všetkými zamestnancami

Obe podmienky spĺňajú len platforma Slack a služba od Microsoftu.

Možnosť API autorizácie na základe účtu v Microsoft Azure, by vyžadovala investíciu spoločnosti v podobe zakúpenia vyššieho balíčku služieb, pretože SBS 2011 túto službu neposkytuje. Nevýhodou je takisto využívanie vlastných počítačov na prácu niektorých zamestnancov, a teda nepracujú pod doménovými účtami.

Zamestnanci využívajú Slack ako desktopovú ale aj mobilnú aplikáciu v rámci celej spoločnosti. Výhodou Slacku je jeho obsiahla dokumentácia a možnosti rozširovania implementácie. Poskytuje rôzne vylepšenia bežnej autorizácie, napríklad autorizácia na základe príslušnosti účtu k danej workspace, alebo na základe oprávnení daného účtu (administrátor / účastník). Takisto možnosť rozšírenia o ďalšie funkcie, ktoré Slack poskytuje. Napríklad automatickej správy (notifikácie) v Slacku po prihlásení do administrácie.

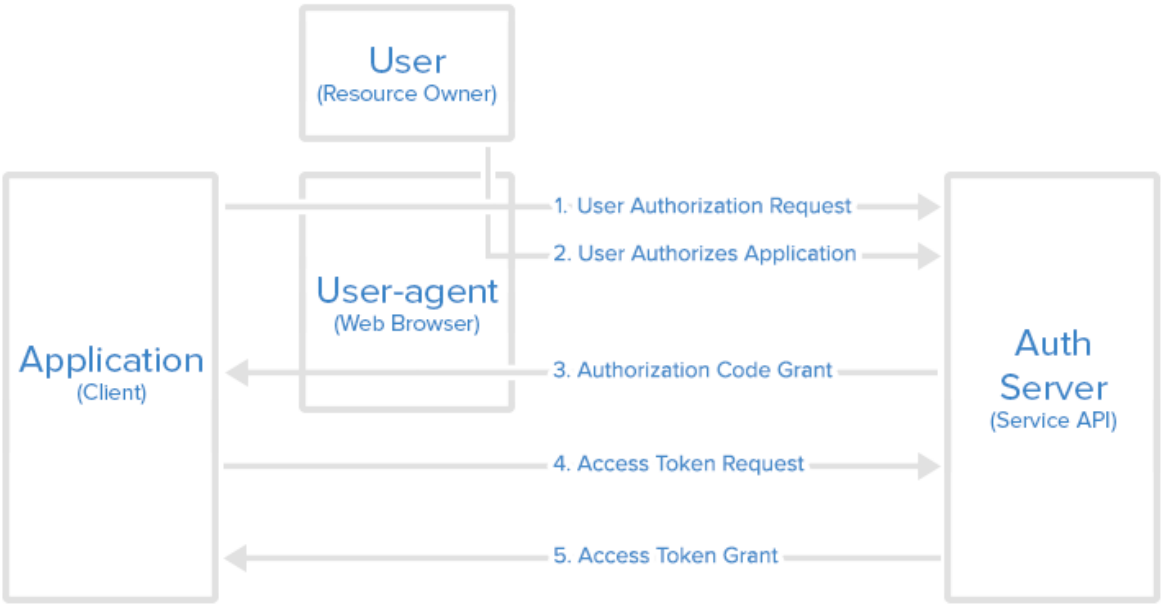
Pre API autorizáciu bola zvolená platforma **Slack.**

3.4. Priebeh autorizačného grantu

Návrh je postavený na protokole OAuth 2.0 a jeho princípe fungovania autorizačného grantu. Ich všeobecné fungovanie a princíp bol popísaný v teoretických východiskách práce. V tejto časti bude priebeh autorizačného grantu implementovaný na konkrétne riešenie, teda na API autorizáciu v platforme Slack.

V danom prípade sú role definované ako:

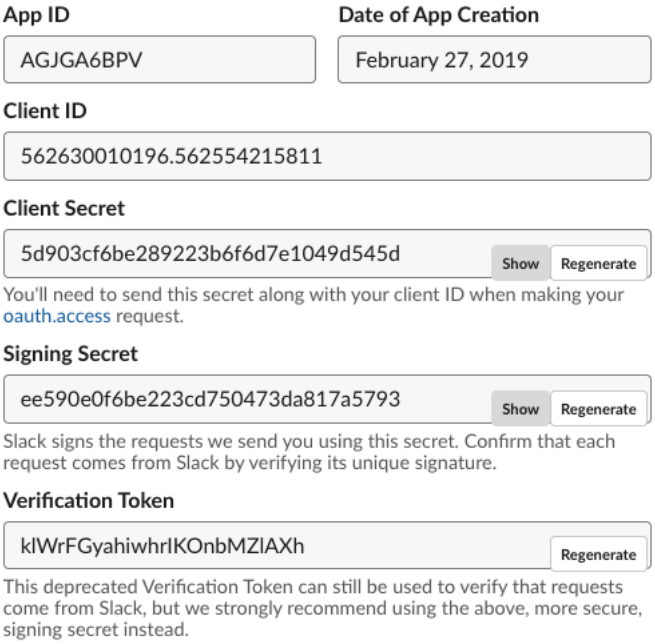
- Klient (**aplikácia**) je konkrétny *CMS systém*.
- Vlastník zdroja (**používateľ**) je *zamestnanec*, ktorý žiada o autorizáciu.
- Autorizačný a zdrojový server (**API**) je platforma *Slack*.



Obrázok 12: Priebeh OAuth 2.0 autorizačného grantu (zdroj: 8)

3.4.1. Krok 0 - Zaregistrovanie aplikácie

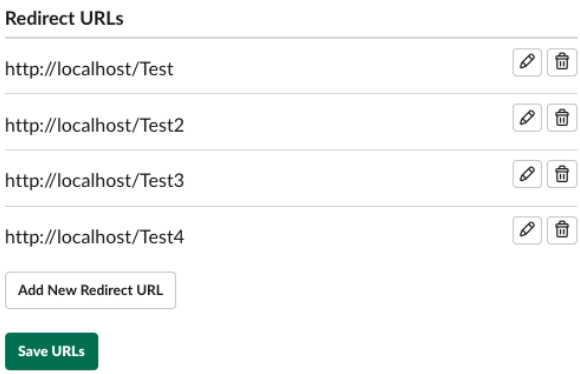
Je nutné zaregistrovať aplikáciu v službe poskytovateľa - Slacku (*https://api.slack.com*). Registrácia aplikácie je na základe existujúceho účtu v platforme Slack. Pre účel práce bol vytvorený workspace s názvom ***Test Diplomka***.



Obrázok 13: Poverenia zaregistrovanej aplikácie v API Slacku (zdroj: 7)

Po zaregistrovaní aplikácie, služba Slacku pridelí „klientské poverenia“ vo forme *Client ID* a *Client Secret*. ID klienta je verejne šíriteľný reťazec, ktorý používa API služba Slacku na identifikáciu aplikácie a tiež sa používa na vytvorenie autorizačných adries URL, ktoré sú prezentované používateľovi. Client Secret sa používa na autentifikáciu identity aplikácie do API služby, keď aplikácia požaduje prístup k používateľskému účtu. Musí byť udržiavaná ako súkromná pri prenose medzi aplikáciou a rozhraním API.

Následne je nutné definovať URL adresy presmerovania. Odovzdaná *redirect_uri* v žiadosti o autorizáciu (nasledujúci krok), sa musí zhodovať s jednou z definovaných adries.



Obrázok 14: Príklad zoznamu URL adries presmerovania v lokálnom prostredí (zdroj: 7)

3.4.2. Krok 1 - Žiadosť o autentifikáciu používateľa

Proces prihlásenia začína žiadosťou užívateľa o autentifikáciu na API Slacku smerovaním na adresu:

https://slack.com/oauth/authorize

spolu s nastaviteľnými GET parametrami žiadosti:

- ***client_id*** - jednoznačné ID pridelené našej aplikácií.
- ***scope*** - rozsah povolení (špecifikujú ako aplikácia požaduje pristupovať k účtu).
- ***redirect_url*** - URL adresa pre presmerovanie.
- ***state*** - jedinečný reťazec, ktorý sa po dokončení predá. Ide o voliteľný parameter a má za úlohu zabrániť útokom s falšovaním hodnôt.
- ***team*** - Slack ID konkrétneho workspace (pracovného prostredia). Ak sa v tomto parametri odovzdá platné ID pracovného prostredia, **v ktorom je užívateľ prihlásený** do daného pracovného prostredia, odovzdanie tohoto parametru zabezpečí, že používateľ len autorizuje daný pracovné prostredie.

Príklad URL adresy s vyžadovanými GET parametrami:

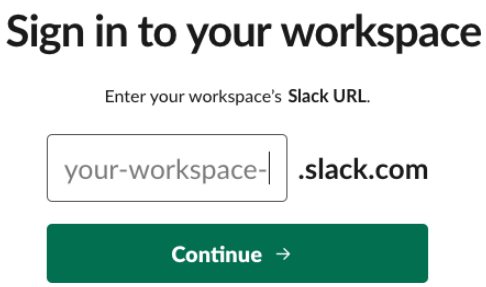
*https://slack.com/oauth/authorize?response_type=**code**&client_id=**CLIENT_ID**&redirect_uri=**REDIRECT_URI**&scope=**identity.basic***

Povinné GET parametre sú *client_id* a *scope*. Parameter *state* je nepovinný, jeho využitie zvyšuje bezpečnosť procesu pred zásahom tretej strany (XSS útoky) a jeho význam a využitie bude v ďalšej práce . Predaním platného parametru *team*, užívateľ nebude nútený zadať názov príslušného workspace (krok 2A).

3.4.3. Krok 2 - Autentifikácia užívateľa

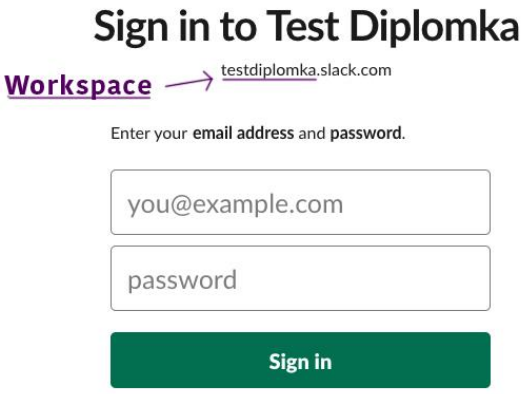
A.) Prvý krok je autentifikácia, teda overenie identity užívateľa. Ak je užívateľ prihlásený v prehliadači vo svojom Slack účte v danom workspace, je tento krok vynechaný a užívateľ pokračuje **bodom B**.

Autentifikácia začína zadaním príslušného pracovného prostredia, do ktorého užívateľ žiada o prístup. Tento krok sa vynechá v prípade, ak žiadosť obsahuje parameter *TEAM* s platným ID daného pracovného prostredia (workspace).



Obrázok 15: Zadanie workspace (parameter Team) (zdroj: 7)

Užívateľ sa overí na základe prihlasovacieho emailu a hesla. V spoločnosti sa na prihlásenie využívajú firemné emailové adresy.



Obrázok 16: Overenie prístupu užívateľa k workspace (zdroj: 7)

Po úspešnej autentifikácií účtu k danému workspace, proces prihlásenia pokračuje bodom **B.**

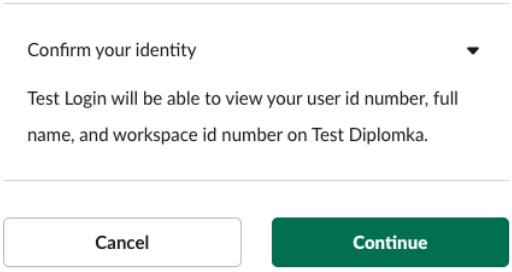
B.) V tomto kroku užívateľ musí autorizovať aplikáciu k prístupu k informáciám z jeho Slack účtu. Rozsah týchto informácií je daný parametrom *Scope*, ktorý špecifikuje ako aplikácia vyžaduje pristupovať k informáciám. Parameter *Scope* sa vzťahuje na objekt, ktorému poskytuje prístup, nasledovaný triedou akcií na danom objekte, ktorý povoľuje (napr. *file: write,* povoľuje aplikácií pridávať, editovať a vymazávať súbory a komentáre k súborom).

Sú definované hlavné triedy akcií:

- **read**: povoľuje čítanie informácií o objekte
- **write**: modifikáciu objektu (úpravu, vytvorenie, vymazanie)
- **history**: prístup k archívu správ kanálov
- **identity**: umožňuje aplikácií potvrdiť vašu identitu
- **client**: umožňuje aplikácií, aby sa odosielali správy v mene používateľa
- **admin**: umožňuje aplikáciám vykonávať administratívne akcie (autorizovaný užívateľ musí byť administrátorom)

Základné bezpečnostné opatrenie je pracovať len s rozsahom informácií o užívateľoch, ktoré sú nevyhnutné pre chod aplikácie. Naša aplikácia potrebuje len overiť identitu užívateľa a jeho workspace. Dostačujúce informácie pre chod aplikácie sú teda *ID a meno užívateľa a ID*

workspace. Parameter *scope* je preto definovaný (na nevyhnutné minimum pre chod aplikácie) ako ***identity.basic*.** Žiadosť našej aplikácie sa užívateľovi zobrazí nasledovne:



Obrázok 17: Autorizovanie aplikácie pre prístup k informáciám z účtu Slack (zdroj: 7)

Príklad odpovede vo formáte JSON pre parameter *scope* definovaný ako ***identity.basic***:


```
{
    "ok": true,
    "user": {
        "name": "Sonny Whether",
        "id": "U0G9QF9C6"
    },
    "team": {
        "id": "T0G9PQBBK"
    }
}
```


Obrázok 18: Príklad JSON odpovede (zdroj: vlastné spracovanie)

3.4.4. Krok 3 - Presmerovanie na server aplikácie

Ak užívateľ súhlasí s prístupom aplikácie k jeho informáciam v účte Slacku, je presmerovaný na adresu *redirect_url*, ktorá je definovaná v nastavení aplikácie (*krok 0*), spolu s GET parametrom *code*. Dočasný **verifikačný parameter** *code*, slúži v nasledujúcom kroku na žiadosť o výmenu za prístupový token.

3.4.5. Krok 4 - Aplikácia požiada o prístupový token

Aplikácia žiada o prístupový token z rozhrania API tým, že odovzdá autorizačný kód spolu s podrobnosťami autentifikácie, vrátane *client_secret*. Príklad žiadosti na koncový bod API Slacku:

*https://slack.com/api/oauth.access/token?client_id=**CLIENT_ID**&client_secret=**CLIENT_SECRET**&grant_type=authorization_code&code=**CODE**&redirect_uri=**REDIRECT_URI***

- *client_id* - pridelené pri vytváraní aplikácie
- *client_secret* - pridelené pri vytváraní aplikácie
- *code* - dočasný autorizačný kód
- *redirect_uri* - musí sa zhodovať s jednou z nadefinovaných URL presmerovania

V prípade **zamietnutia** požiadavky, Slack presmeruje späť na *redirect_uri* spolu s GET parametrom error.

```
http://yourapp.com/oauth?
  error=access_denied
```

Obrázok 19: Zamietnutie autorizácie (Zdroj: 7)

V prípade úspešnej autorizácie, je obdržaná JSON odpoveď, ktorá obsahuje nasledujúce pole hodnôt:

```
{
    "ok": true,
    "access_token": "xoxp-1111827399-16111519414-20367011469-5f89a31i07",
    "scope": "identity.basic",
    "team_id": "T0G9PQBBK"
}
```

Obrázok 20: JSON odpoveď obsahujúca access_token a scope parameter (Zdroj: 7)

Aplikácia je autorizovaná a môže použiť token na prístup k používateľskému účtu prostredníctvom API služby, ktorý je obmedzený na rozsah prístupu, ktorý definuje parameter *scope*.

3.5. Nadstavba pre viac aplikácií

V tejto časti sú popísané dve možnosti ako proces autorizácie rozšíriť pre potreby spoločnosti a teda **pre autorizáciu viac aplikácií** (CMS systémov).

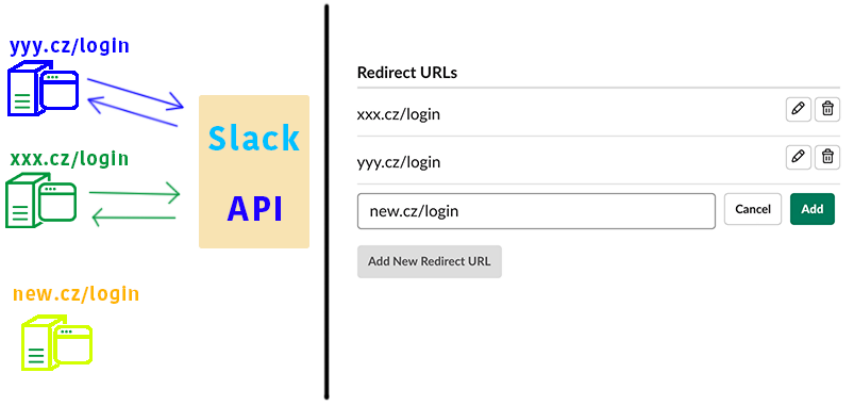
Aplikácia (CMS systém) úspešnou autorizáciou užívateľa, dostala potrebné údaje na autorizáciu prístupu (ID, meno a ID workspacu) spolu s prístupovým tokenom. Ide teda o autorizáciu medzi jedným klientom (aplikáciou) a API Slacku. Spoločnosť však pracuje s niekoľko desiatkami CMS systémov (aplikácií) na rôznych hostingových priestoroch. Každý CMS systém beží na samostatnej URI adrese a teda predstavuje aj samostatnú adresu presmerovania. Konfigurácia zoznamu adries presmerovania (*redirect_uri*) bola ukázaná pri registrácií aplikácie v API Slacku (*krok 0*).

Parameter *redirect_uri* je odovzdaný pri žiadosti o autorizáciu užívateľa (*krok 1*) *GET* metódou. Ak je tento parameter predaný v žiadosti (krok 1) o autorizáciu, musí byť zhodný s jednou z adries presmerovania definovaných v zozname v nastavení aplikácie (*krok 0*).

Parameter však nie je povinný a teda ak je v nastavení aplikácií definovaná len jedna adresa presmerovania, proces bude automaticky brať ju aj bez predania jej hodnoty v GET parametre *redirect_url*.

3.5.1. Rozširovanie zoznamu Redirect URLs

Prvý možný spôsob riešenia nadstavby autorizácie pre viac aplikácií je možný rozširovaním adries zoznamu presmerovaní v nastavení aplikácie (*krok 0*). Pri spustení nového CMS systému, je teda nutné zaregistrovať danú adresu presmerovania v nastaveniach aplikácie.



Obrázok 21: Riešenie Redirect URLs zoznamom (zdroj: vlastné spracovanie)

Výhodou riešenia rozširovaním zoznamu je jednoduchosť implementácie, nevyžaduje žiadne ďalšie konfigurácie aplikácie.

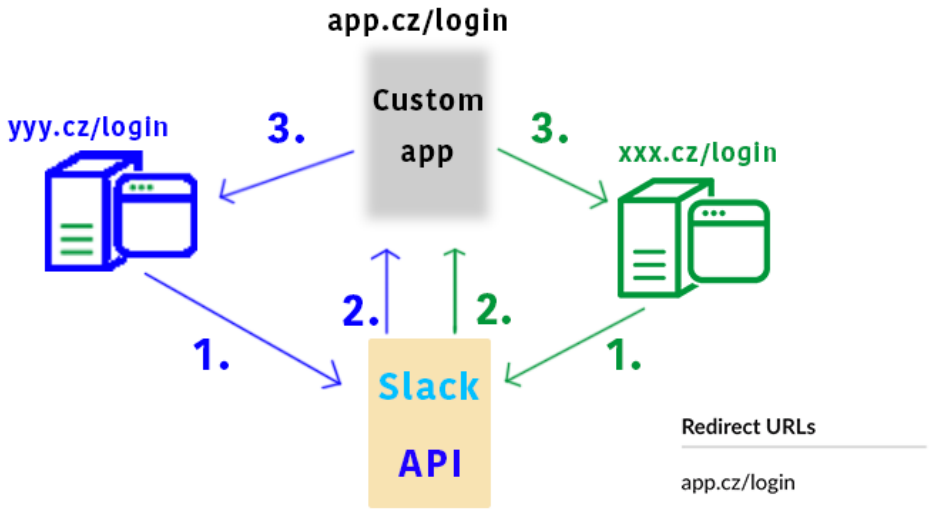
Jej hlavnou nevýhodou je neprehľadnosť a ťažké riadenie. V zoznamoch je nemožné vyhľadávanie a filtrovanie zoznamu podľa kritérií. Neexistuje limit pre maximálny počet adries v zozname, pri vysokom počte však proces autorizácie spomaľuje (loading time). Testovaním bolo zistené spomalenie o 1,02 sekundy pri počte 100 adries presmerovania oproti rýchlosti načítania autorizačných stránok Slacku pri jednej adrese presmerovania.

Klady	Zápory
+ Jednoduchá implementácia	- Neprehľadnosť - Spomalenie procesu autorizácie

Riešením spomalenia autorizačného procesu zapríčinený vysokým počtom adries presmerovania v zozname aplikácie, by bolo vytvorenie viacerých samostatných aplikácií, do ktorých by boli adresy presmerovania rozdelené. To však komplikuje proces implementácie riešenia do produkčného prostredia a takisto zhoršuje kontrolu nad správou týchto adries. Zo spomenutých dôvodov tento spôsob nebude využitý

3.5.2. Centrálna aplikácia pre riadenie procesu autorizácií

Druhou možnosťou je vytvorenie centrálnej aplikácie pre riadenie procesu autorizácie. V nastavení aplikácie je teda potrebné mať len jednu adresu presmerovania.



Obrázok 22: Princíp centrálnej aplikácie (zdroj: vlastné spracovanie)

V nastavení aplikácie (*krok 0*) je definovaná len jedna adresa presmerovania pre všetky CMS systémy, a preto nie je nutné v žiadosti o autorizáciu (krok 1) uvádzať GET parameter `redirect_uri`. Je však nutné predať informáciu o tom, do ktorého CMS systému užívateľ žiada o autorizáciu. Respektíve, z ktorého CMS systému prišla žiadosť do API Slacku. Následne po úspešnom overení prístupov mu prideliť práva len pre danú administráciu. Je teda nutné predať centrálne aplikácií (*custom app*) informáciu o tom, aká URI je pôvodným žiadateľom. Aplikácia dostáva autorizačný kód od API Slacku (HTTP referrer), ktorý vymení za autorizačný token. V prípade úspešnej výmeny autorizačného kódu za prístupový token, aplikácia potrebuje poveriť príslušnú administráciu o povolení prístupu. Musí teda vedieť, aká URI bola pôvodným žiadateľom.

Riešením je parameter `state`. Ide o nepovinný parameter žiadosti o autorizáciu (*krok 1*). Aplikácia spolu s autorizačným kódom (`code`) obdrží parameter `state` na základe ktorého určí príslušnú administráciu. Parameter `state` môže obsahovať ľubovoľný počet znakov a jeho formát môže byť volený náhodnými znakmi. Centrálna aplikácia teda potrebuje uchovávať v databázy URL adresy pre príslušné `state` parametre.

Stĺpec	Typ
id	int(11) *Auto Increment*
state	varchar(40)
url	varchar(50)
key	varchar(50)

Príkladom takéhoto návrhu tabuľky môže vyzerať nasledovne.

id	state	url	key
1	abCdEfgH	xxx.cz/login	xTraGrtoPa1As521Dad
2	iJklMpRst	yyy.cz/login	xUpiTrupka45Trakulita
3	tRoPklasD	new.cz/login	xTra8M0QurLaadPio7Tr

Príklad URL žiadosti používateľa na autorizáciu do administrácie z adresy yyy.cz/admin by spolu s GET parametrami:

```
https://slack.com/oauth/authorize?response_type=code&client_id=CLIENT_ID&state=iJklMpRst&scope=identity.basic
```

Parameter `redirect_uri` nie je nutné uvádzať. V nastavení aplikácie je konfigurovaná jediná adresa presmerovania *app.cz/login*, a teda je automaticky nastavená.

Nasleduje proces autorizácie medzi centrálnou aplikáciou a API Slacku (*kroky 2, 3 a 4*). Čoho výsledkom je že centrálna aplikácia obdrží prístupový token spolu s parametrom `state` na základe ktorého musí prideliť používateľa prístup k danej administrácií. Predanie prebehne na základe obdržania aplikácie JSON Web Tokenu.

3.5.3. Vytvorenie JSON Web Tokenu

K predaniu prístupu z centrálnej aplikácie do konkrétneho CMS systému je využitý JWT. V tejto časti bude popísané vytvorenie JWT generátoru v jazyku PHP.

Vytvorenie Header a Payload

Pre vytvorenie hlavičky a payloadu je potrebné vyrobiť JSON reťazce. V hlavičke je definovaný `typ` a `alg` (algoritmus kódovania). V payloade je vytvorené vlastné pole `user_id`.

```
// Vytvorenie token header ako JSON string
$header = json_encode(['typ' => 'JWT', 'alg' => 'HS256']);

// Vytvorenie token payload ako JSON string
$payload = json_encode(['user_id' => 123]);
```

V uvedenom príklade je vytvorená hlavička a payload, pričom predaný parameter user_id je konštanta 123, v príklade sa nepracuje priamo s premennou, ktorá využíva metódu na dotaz priamo na databázu.

Base64Url kódovania Header a Payload reťazcov

Base64 je dátový formát zobrazujúci binárne dáta pomocou ASCCII znakov. Base64Url vychádza z Base64 kódovania, okrem toho že nevyužíva rezervované URL znaky (+, /, =).

Zakódovanie premenných `$header` a `$payload` JSON reťazov na **Base64Url** reťazce. Metóda Base64Url nie je v PHP vytvorená a je nutné nahradiť znaky + za -, / za _ a = za ''. To zabezpečuje aby nebola potrebné ďalšie URL kódovanie.

```
// Kódovanie Header do Base64Url reťazca
$base64UrlHeader = str_replace(['+', '/', '='], ['-', '_', ''], base64_encode($header));

// Kódovanie Payload do Base64Url reťazca
$base64UrlPayload = str_replace(['+', '/', '='], ['-', '_', ''], base64_encode($payload));
```

Vytvorenie podpisu

Pri vytvorení podpisu je využitá metóda hash_hmac(), ktorá je dostupná v jazyku PHP a použitý algoritmus sha256. Premenné `$base64UrlHeader` a `$base64UrlPayload` sú oddelené bodkou, čo je štandardný zápis JWT. Vstupom pre hash_hmac metódu je aj tajný kľúč: `abC123!`. V produkčnom prostredí je vhodné minimálnu dĺžku tajného kľúča nastaviť na 12 znakov.

```
// Vytvorenie Podpisu
$signature = hash_hmac('sha256', $base64UrlHeader . "." . $base64UrlPayload, 'abC123!', true);
```

Uvedený príklad vytvárania podpisu pracuje s konštantou, nevyužíva metódu na dotaz do databázy na parameter *key*.

Base64Url zakódovanie podpisu

Vytvorený podpis je nutné zakódovať do Base64Url takisto ako Header a Payload.

```
// Encode Signature to Base64Url String
$base64UrlSignature = str_replace(['+', '/', '='], ['-', '_', ''], base64_encode($signature));
```

Vytvorenie JSON Web Tokenu

Spojením premenných `$base64UrlHeader`, `$base64UrlPayload` a `$base64UrlSignature` je vytvorený JWT.

```
// Vytvorenie JWT tokenu
$jwt = $base64UrlHeader . "." . $base64UrlPayload . "." . $base64UrlSignature;

// Výstup
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoxMjN9.NYlecdiqVuRg0XkWvjFvpLvglmfR1ZT7f8HeDDEoSx8
```

Takto vytvorený JWT token je odoslaný na URL adresu príslušnej administrácie definovanej podľa parametru `stat`.

3.5.4. Spracovanie tokenu aplikáciou

Klient (konkrétny CMS systém) ktorý obdržal od centrálnej aplikácie takto vytvorený token, sa autentifikuje na základe hodnoty tajného kľúča. Vytvorením session aplikácie v prehliadači sa poskytne prihlásenie pre používateľa, ktorý je identifikovaný na základe parametru `user_id` ktorý je obsahom payloadu prijatého JWT tokenu.

3.6. Zásady bezpečnosti

JWT token nezabezpečuje šifrovanie prenášaného obsahu (payload), len jeho jednoduché kódovanie, ktoré je ľahko čitateľné. V tejto časti teda nemožno prenášať žiadne citlivé dáta. JWT token zabezpečuje, že jeho obsah nejde zmeniť, jeho obsah je však čitateľný aj pre tretie strany. Šifrovaný je len podpis, ktorý slúži na autentifikáciu serveru a overenie, že informácie v payloade neboli upravené počas prenosu. Podpis je nutné overovať vždy pred spracovaním tokenu. Vhodným bezpečnostným opatrením je rozšírenie payloadu o čas expirácie nastaveným na niekoľko minút. Pri prenose je nutné využiť HTTPS protokol, ktorý zabraňuje útokom "Man in the middle".

Pri udržiavaní dôveryhodných poverení (*access_token*) je základný opatrením nepracovať s tokenmi získaných metódou GET prostredníctvom URL adresy. Pri prenášaní prístupového tokenu medzi API Slacku a centrálnou aplikáciou, je nutné použiť metódu POST. Pri najvyšších vrstvách je nutné myslieť na útoky ako XSS, CSRF, SQL injection.

Príklad ochrany pred CSRF útokom je kontrola predaného GET parametru `state`, ktorý predstavuje jedinečný reťazec znakov. V prípade ak nie je zadaný, ide zrejme o útok tretej strany.

```
// Kontrola GET parametru state pred CSRF útokom
if (empty($_GET['state']) || ($_GET['state'] !== $_SESSION['oauth2state'])) {

    unset($_SESSION['oauth2state']);
    exit('Invalid state');

}
```

Obrázok 23: Kontrola parametru GET proti CSRF útoku (zdroj: vlastné spracovanie)

3.7. Odporúčanie pre spoločnosť k implementácií riešenia

Cieľom doporučení je snaha k minimalizovaniu neočakávaných problémov pri implementácií návrhu do produkčného prostredia spoločnosti. S tým spojená nedostupnosť administrácií v čase, kedy klient potrebuje prístup.

Realizovať proces prechodu na nový systém prihlasovania v niekoľkých etapách. Prvá etapa predstavovala vývoj a testovanie v lokálnom prostredí, bez produkčných databáz a CMS systému spoločnosti. Ďalším krokom je implementácia a testovanie vo vývojovom prostredí spoločnosti s využitím vývojových databáz a implementáciou priamo na CMS systém spoločnosti.

Po úspešnej implementácií a testovaní vo vývojovom prostredí spoločnosti, zahájiť postupný prechod do produkčného prostredia. Začať implementáciou v jednom CMS systéme. Vybrať systém, ktorý spracováva minimálny rozsah dát s nízkou prevádzkovou návštevnosťou. Aby dočasný výpadok administrácie, spôsobený neočakávanými problémami pri implementácií, nespôsobil problémy klientovi a tým aj spoločnosti. Prípadná implementácia v nočných hodinách, znižuje riziko s odstávkou administrácie v čase kedy klient potrebuje prístup.

Po úspešnej implementácií odstaviť súčasný spôsob prihlasovania a zapojiť zamestnancov do testovania s ich spätnou väzbou.

Po uplynutí doby testovania, pokračovať v postupnej implementácií smerom k väčším projektom.

3.7.1. Odporúčania pre spoločnosť v bezpečnosti

Riešenie autorizácie zamestnancov v administráciách klientov na základe účtu v systéme tretej strany, poskytuje niekoľko bezpečnostných zlepšení oproti súčasnému stavu. Je však nutné dodržiavať zásady pri uchovávaní hesla od účtu Slacku, na základe ktorého je postavená autorizácia do všetkých administrácií. Zásady bezpečnosti hesiel, ako napríklad neprihlasovať sa do účtu na cudzích zariadeniach, neuchovávať heslá v nešifrovanej forme a periodická aktualizácia hesla. Riziká spojené so slabým prihlasovacím heslo rieši Slack, ktorý vyžaduje minimálnu dĺžku a kombináciu znakov.

Pred implementáciou nového riešenia je vhodné ozrejmiť zodpovednosť zamestnancov za ich prácu s prístupovými údajmi do Slacku. A dopad v prípade zneužitia ich prístupov, kedy sú v ohrození všetky klientské administrácie.

3.8. Vývoj do budúcna

Prihlásenie postavené na API Slacku poskytuje možnosť rozširovať návrh. Možnosť zvýšenia zabezpečenia by mohlo implementácie Slack API bot messengera, ktorého je možné nakonfigurovať na odosielanie správ po určitej udalosti. Tou by mohlo byť napríklad prihlásenie do CMS systému. Zamestnanec by dostal notifikačnú správu po prihlásení a tým by bol okamžite informovaný o prípadnom neoprávnenom prihlásení jeho účtu do CMS systému.

Zlepšenie v procese registrovania nového CMS systému do procesu autorizácie, by predstavovalo vytvorenie webového rozhrania pre vkladanie novej administrácie - teda jeho `scope` a `redirect_url` parametrov. Programátor by teda nemusel zasahovať priamo do databázy, ale vo webovom rozhraní by bol schopný zaregistrovať novú administráciu. Týmto by sa zamedzilo neúmyselným vymazaním údajov z databázy.

3.9. Prínosy riešenia

Hlavný prínos nového riešenia je v zvýšení úrovne zabezpečenia dát. Zmenou systému prihlasovania zamestnancov do CMS systémov, sa eliminovali najväčšie bezpečnostné hrozby spojené s neoprávneným prístupom k dátam klientov.

Prínosy v porovnaní so súčasným spôsobom prihlasovania jedným účtom pre všetkých zamestnancov a novým riešením autorizácie na základe účtov zamestnancov v Slacku:

Ukončovanie prístupov zamestnancov

Autorizácia prebieha na základe Slack účtu v pracovnom prostredí spoločnosti (workspace), z ktorého je zamestnanec automaticky odstránený pri ukončení pracovného pomeru. A teda stráca prístupové práva do CMS systémov spoločnosti.

Prihlasovanie vlastným účtom

Identita zamestnanca je pradená na základe jeho user_id v Slack API. Na základe tohoto parametru je užívateľ identifikovaný a tým preberajá zodpovednosť za prácu v administráciách. Čo zlepšuje sa kontrolu práce zamestnancov.

Zamestnanci v súčasnom spôsobe majú spoločné heslo, ktoré je vhodným terčom útokom sociálneho inžinierstva (napríklad pretexting).

Jeden prístupový účet

V súčasnom spôsobe je pre každý CMS systém rozdielne heslo. S veľkým počtom CMS systémov, ktoré má spoločnosť v súčasnosti v prevádzke, vznikajú riziká spojené s nesprávnym uchovávaním týchto hesiel. V nezašifrovanom formáte v počítači ale aj formou poznámok na papieri.

3.10. Ekonomické zhodnotenie

Prínos je v oblasti bezpečnosti a neprináša priamy zisk pre spoločnosť, ale zvyšuje zabezpečenie pred ekonomickým zaťažením v podobe zmluvných pokút. V zmluve s klientom sa uvádzajú pokuty v prípade úniku dát klienta, pričom ich výška je niekoľko miliónov českých korún. Výška závisí od veľkosti projektu, respektíve rozsahu informácií ktoré daný CMS systém spracováva.

Náklady riešenia sú spojené s implementáciou nového spôsobu prihlasovania. Časový odhad je uvedený v človekohodinách. Implementácia je riadená a vykonaná vnútri spoločnosti. Žiadne ďalšie náklady, v prípade využitia firemného serveru pre centrálnu aplikáciu, nové riešenie neprináša.

Tabuľka 13: Ekonomické náklady riešenia (zdroj: vlastné spracovanie)

Činnosť	Trvanie
Analýza a výber riešenia	25
Konzultácie s vedením	8
Lokálny vývoj	24
Implementácia do CMS systému	16
Testovanie	4
Nasadenie do produkcie pre vybraný CMS systém	3
Testovanie na produkčnom prostredi	3
Rozšírenie do všetkych CMS systémov	8
SPOLU	**91**

Náklady na nový spôsob prihlasovania zamestnancov do CMS systémov boli odhadnuté na 91človekohodín, pričom zvyšujú zabezpečenie pred zmluvnými pokutami vo výške niekoľkých miliónov českých korún.

ZÁVER

Cieľom diplomovej práce bol návrh a implementácia riešenia s cieľom zvýšenia bezpečnosti dát spoločnosti. Z uskutočnených analýz bolo zistených niekoľko bezpečnostných slabín spojených zo súčasným spôsobom prihlasovania zamestnancov do CMS systémov. Nesprávne ukončovanie prístupových práv a nedostatočné zabezpečenie systémov boli hlavnými podkladmi pre návrhovú časť.

V hlavnej časti práce - návrh vlastného riešenia, je postup návrhu a implementácie nového spôsobu prihlasovania zamestnancov. Prvým krokom bolo rozhodnúť o novom spôsobe autorizácie z troch možných riešení. Prvé dva spôsoby predstavujú rozšírený štýl, ktorý je využívaný spoločnosťami v tomto obore. Tretí spôsob bol navrhnutý na základe analýz nových možností prihlasovania so snahou minimalizovať administratívnu záťaž. Následne bol zvolený systém tretej strany, ktorý poskytuje verejnú API autorizáciu. Bol rozobratý priebeh autorizačného protokolu OAuth 2.0 v konkrétnom prostredí API Slacku. Priebeh autorizačného grantu bolo nutné rozšíriť pre autorizáciu viacerých aplikácií (CMS systémov). Riešením je centrálna aplikácia, ktorá riadi proces autorizácie pre všetky CMS systémy spoločnosti a prideľuje oprávnenie prístupu pre konkrétnu administráciu.

V poslednej časti je odporúčanie pre spoločnosť pri implementácií a v bezpečnosti nového spôsobu prihlásenia. Ďalej sú zhrnuté prínosy práce, ktoré sú hlavne z oblasti dátovej bezpečnosti. V závere je odhad nákladov pre spoločnosť na implementáciu riešenia.

ZOZNAM POUŽITÝCH ZDROJOV

(1) MOLNÁR, Zdeněk. *Efektivnost informačních systémů*. Praha: Grada Publishing, spol. s.r.o., 2000. ISBN 80-7169-410-X

(2) SODOMKA, Petr a Hana KLČOVÁ. *Informační systémy v podnikové praxi*. 2. aktualiz. a rozš. vyd. Brno: Computer Press, 2010, 501. ISBN 978-80-251-2878-7.

(3) BASL, J. a R. BLAŽÍČEK. *Podnikové informační systémy: Podnik v informační společnosti.* 3. aktualiz. a dopl. vyd. Praha: Grada, 2012. ISBN 978-80-247-4307-3.

(4) MLÝNEK, Jaroslav. Zabezpečení obchodních informací. Brno: Computer Press, 2007, vi, 154 s. : il. ISBN 978-80-251-1511-4.

(5) DOSTÁL, Petr, Karel RAIS a Zdeněk SOJKA. *Pokročilé metody manažerského rozhodování: konkrétní příklady využití metod v praxi*. Praha: Grada, 2005, 166 s. : il. ISBN 80-247-1338-1.

(6) KOCH, Miloš. ZEFIS – Výzkumný portál Ústavu informatiky Fakulty podnikatelské VUT v Brně [online]. 2013 [cit. 2013-03-26]. Dostupné z: www.zefis.cz

(7) Slack. *The collaboration hub* [online]. [cit. 2019-04-04]. Dostupné z: https://www.api.slack.com/

(8) OAuth 2.0. An open protocol to allow secure authorization [online]. Dostupné z: https://developer.okta.com/blog/2018/04/10/oauth-authorization-code-grant-type

(9) GÁLA, Libor, Jan POUR a Zuzana ŠEDIVÁ. *Podniková informatika*. 2., přeprac. a aktualiz. vyd. Praha: Grada, 2009, 496 s. : il. ISBN 978-80-247-2615-1.

(10) What Is The Difference Between A URI And A URL? [online]. 2019 [cit. 2019-04-04]. Dostupné z: https://dev.to/flippedcoding/what-is-the-difference-between-a-uri-and-a-url-4455

(11) Social Engineering and Reverse Social Engineering [online]. 2018 [cit. 2019-04-04]. Dostupné z: http://www.ittoday.info/AIMS/DSM/82-10-43.pdf

(12) GROSSMAN, Jeremiah. *XSS attacks: cross site scripting exploits and defense*. Burlington: Syngress, 2007, xiv, 448 s. ISBN 1-59749-154-3.

(13) ANDREU, Andres. *Professional pen testing for web applications*. Indianapolis: Wiley, 2006, xxiv, 522 s. ISBN 978-0-471-78966-6.

(14) AMUNDSEN, Mike, Sam RUBY a Leonard RICHARDSON. RESTful Web APIs. 1. O'Reilly Media, 2013. ISBN 9781449358068.

(15) BECHER, Michael. Web application firewalls: applied web application security. Saarbrücken: VDM Verlag Dr. Müller, 2007, iv, 153 s. ISBN 978-3-8364-0446-4.

(16) MATTHEWS, Jeanna. *Computer networking: internet protocols in action*. Hoboken: John Wiley, 2005, xii, 273 s. : il. + 1 CD-ROM. ISBN 978-0-471-66186-3.

(17) STEVENS, Richard. TCP/IP iIllustrated / část I-III. Boston: Addison-Wesley, 1994, 576 s. ISBN 0-201-63346-9.

(18) DOSEDĚL, Tomáš. Počítačová bezpečnost a ochrana dat. Brno: Computer Press, 2004, 190 s. ISBN 80-251-0106-1.

(19) MILLER, Philip M. *TCP/IP: the ultimate protocol guide vol. 2 ; applications, acces and data security*. Boca Raton: BrownWalker Press, 2009, xxxi, s. 533 - 1009. : il. ISBN 978-1-59942-493-4.

(20) STEFANOV, Stoyan a Kumar Chetan SHARMA. Object Oriented JavaScript. Olton: Packt Publishing, 2013. ISBN 9781849693127.

(21) HRISTOZOV, Kasimir. Create and Verify JWTs in PHP [online]. Dostupné z: https://developer.okta.com/blog/2019/02/04/create-and-verify-jwts-in-php

ZOZNAM POUŽITÝCH OBRÁZKOV

ZOZNAM POUŽITÝCH TABULIEK